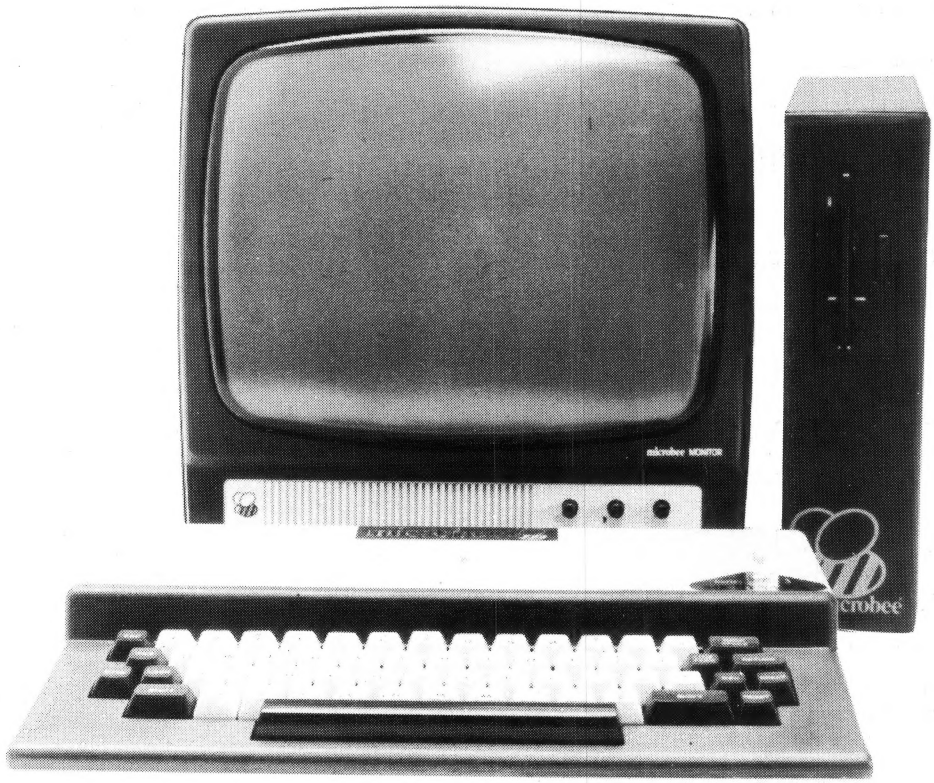
May 1985
$ 0.00
NZ $ 0.00

Bonus Book Of Priceless Programs

Amiga Format

The microbee Computer-in-a-Book represents a novel, practical effective low cost way of advancing to disk drive capability



The microbee 3.5 inch 400K high speed/high density Drives are very compact, extremely tough and represent the best value-for-money 'Real Disk Drive' on the market today. The Computer-in-a-Book comes complete with all the software you will need to get started including demo program, MW Basic, Wordbee Wordprocessing, Telcom communications and CP/M with utilities.

Microbee 64K with Single Disk Drive **$995** incl. Tax
Add-on Disk Drive in matching volume **$345** incl. Tax
Ask about Modems, Printers and Business Software.

microbee computer

Designed and manufactured in Australia by Applied Technology

PHONE ORDERS bankcard ACCEPTED

microbee technology centres

N.S.W.
1 Pattison Ave, Waitara 2077
Phone (02) 487 2711

VIC.
729 Glenferrie Rd, Hawthorn 3122
Phone (03) 819 5288

W.A.
141 Stirling Highway, Nedlands
Phone (09) 386 8289

S.A.
151 Unley Rd, Unley 5061,
Phone (08) 272 1384

QLD
455 Logan Rd, Stones Corner, 4120
Phone (07) 394 3688

FACTORY
Koala Crescent, West Gosford 2250
Phone (043) 24 2711

LEDGERMASTER

BY LINDSAY R. FORD
Dreamcards Software

To start our 32-page section of bumper-to-bumper pocket programs, we've purchased the rights to print details of this commercial package written by Lindsay Ford. It'll take a bit of typing ...

WITH TAXATION time almost here again, many readers will be faced with the daunting prospect of having to shuffle through bundles of old receipts and cheque butts to put together their tax returns. This problem is a particular burden to people like me, who run small businesses that don't make enough to justify paying the fees of a good tax accountant. Club treasurers facing the task of preparing their annual reports will also know just what I mean!

One solution is to invest in a software accounts package, but these are generally disk-based, expensive to buy and more sophisticated than most of us need. LedgerMaster was written to overcome these problems. It is a financial database program designed to run on 32 Kbyte (or larger) Microbee computers and has all the features you're likely to need to prepare a profit and loss statement for taxation purposes or for publication in your club magazine. It allows you to create a ledger containing up to 320 transaction records ('files'), which can be stored on tape, printed, sorted and searched as with any general database.

Where LedgerMaster stands out, however, is it also conducts a balance of the ledger files (and of any other ledgers you have 'chained' to it) and compiles these into a proper profit and loss statement. This takes almost the effort out of preparing tax returns or treasurers' reports, whether you're typing in the receipts and payments in one frenzied, last-minute burst or simply tallying a ledger you've created through the year as transactions occurred.

Program Entry

The program is really a compromise between efficient code and legibility. Ideally, a BASIC program needs as many instructions as possible crammed into each line, if maximum speed and economic memory use are to be achieved, but this would have made LedgerMaster extremely difficult to read. I've opted for slightly less-than-optimum line lengths, but even so the size of the program means you have to be extremely careful as you type it in. Please bear the following points in mind:

- Use AUTO mode to enter the program, as this minimises the risk of 'non-printing characters' being accidentally included in the code. If you have my BASIC Screen Editor fitted to your Bee, use it. This will avoid the problem entirely.
- REMs that appear in a line of code (for example in line 1) should be left out entirely.
- Where a line contains a REM but no BASIC code (for example lines 19 and 20), the REM should be entered, but any text following it omitted.
- The up-arrow character (↑) appears frequently in the program. Don't leave it out.
- Watch out for variables I, O and Q. These are easy to confuse with ones and zeros.

Once the program has been entered exactly as shown in the listing (subject to the above comments), use the GX command to locate all lines containing REMs. These lines should be deleted (using the DELETE command) and several copies of the program should then be stored on tape.

At this stage you have an almost complete program. Now type in the following (in 'direct mode' – don't give it a line number):

```
CLS: X=0: Y=PEEK(2258) + 256*PEEK(2259): FOR Z=2304 TO Y:
X=X+PEEK(Z): NEXT Z: PRINT "X="; X, "Y="; Y: END
```

When you press the <RETURN> key the screen will clear, and after a few minutes two numbers will appear. 'Y' is the address in decimal of the top of the BASIC file and should be 11819. If it's higher than this you've put something in the program that shouldn't be there, if lower you've left something out. 'X' is a checksum — it is the total of the numbers contained in each of the memory bytes making up the program. If it is -16146 then all is well and you can proceed to the next stage; otherwise you'll have to look for your typing error.

Stage 2 of the operation involves removal of unnecessary spaces in the listing, as these greatly reduce execution speed. Enter "GX//'"<Return> and rest a brick on the full-stop key, then go and make yourself a cup of coffee. After about five minutes all spaces will have been removed. Now you'll have to insert spaces in the places they're meant to be by substituting a space for each up-arrow. Use "GX/↑//'"to do this. Once done you should save a few copies on tape, then enter the test routine shown above. This time Y should be 10079 and X, 17639. If they are you've got a fully operational LedgerMaster.

Using the Program

When you RUN LedgerMaster there will be a brief delay as the main arrays are initialised, then a menu will appear giving you the option to 'Create Base' or 'Load'. You will also see two numbers at the bottom right of the screen. The first of these is the free memory counter, which tells you how much room remains in the ledger (you can't run out of file space on a 32 Kbyte machine), and the second is the number of free files available. As the ledger hasn't been initialised this second number will be zero. You won't have a ledger tape you can load at this stage, so let's look at the first option.

Create Base

This routine initialises the ledger, clearing out any existing material in memory. For this reason you will be asked if you are 'Sure?' if you access it when a ledger is already present. You will then be requested to name the ledger (maximum 24 characters), then to input the various categories of payments and receipts you want included on the final balance sheet (such as 'insurance', 'rent', 'royalties', 'wages' and so on). Give these a good deal of thought before starting, as they comprise the framework around which your financial report or tax return will be built.

You can enter up to 20 classes of payments and 20 of receipts (maximum 12 characters), and a 'carried forward' balance for each. This balance will be of importance if you are including amounts accrued from a previous period, such as debts outstanding from the last tax year. No 'carried forward' balance will be entered if you press the <Return> key. Pressing <Return> when you're prompted to input a payment or receipt category indicates you've finished entering items in that class.

Once the payments and receipts categories are entered your ledger 'base' is complete and the program will return to the menu. You will now have four options available — 'Create', 'Load', 'Enter' and 'Save'. If you ▷

want to go on and input a couple of files, the first option to look at is 'Enter'.

Enter

Pressing key 3 clears the menu from the screen and the prompt at bottom left asks you to input the 'day' part of the date of the particular file. By now you'll have noticed the program uses a non-standard input routine, the length of the particular field being indicated by a row of star (*) characters next to the prompt. As each character is typed it replaces one of the stars, the entry being complete either when the <RETURN> key is pressed or when the last character in that field is typed.

With date entries the data field is two characters wide, so a date comprising one digit (for example the '1' in 1/12/84) can be entered either as '1 <RETURN>' or as '01'. Once the day is entered you will be prompted for the month and then the year. To avoid unnecessary typing, the program allows you to 'borrow' date data from the last file you typed. Thus, if the date on the last record was '14/3/84' and you press the <RETURN> key in answer to the 'day' prompt in the next record, it will be assigned the date '14/3/84' as well. This can also be used with parts of dates, so if you input '28' for the day, then press <RETURN> when prompted to input the month, the date will be entered as '28/3/84'.

An error check routine is incorporated to detect illegal dates (including February 29 on non-leap years). Note, as each field is entered the partially complete file is displayed at the top of the display window.

After the date is entered you are asked to enter 'From/To'. This identifies from whom the money was received or to whom it was paid (maximum 16 characters). Pressing the <RETURN> key will enter the word 'Paid' in this field. The next item requested is the receipt or invoice number (maximum seven characters), which can be omitted by pressing the <RETURN> key, and the word 'B/card' inserted by pressing ↑ B (that is, the <CONTROL> and 'B' keys simultaneously) or the word 'Cash' by pressing ↑ C.

The next two steps deal with the category into which the item will be inserted in the balance sheet. The first asks you if it is a receipt or payment (press 'R' or 'P'), then all categories included in the ledger base will appear in the display window, with letters to identify them. Press the key corresponding to the particular category into which the payment falls. If you press a key that doesn't correspond to one of the displayed categories it will be ignored.

The final item to enter is the amount of money involved. Although this field is nine characters wide, two spaces are reserved for cents and one for the decimal place. (If you're dealing with amounts of a million dollars or more you can afford to get an accountant to do it for you!) There's no need to input trailing zeros in the cents part, so '100 dollars' can be entered as '100 <RETURN>' and '87 dollars 30 cents' as '87.3 <RETURN>'. Payments and receipts appear in separate right-justified columns in the display window: payments to the right and receipts to the left.

This completes your file entry and you will now be asked whether you wish to proceed to enter the next file (press <RETURN>), to return to the menu (key 'M') or to 're-do' (correct) the record you just typed (key 'R'). The program will not allow you to exceed the ledger capacity, so once the 320th file is typed it will return to the menu and announce 'Ledger Full'.

Save

This option allows you to save all or part of your ledger on tape. Pressing key 4 will take you into the 'SAVE' routine and you'll then be asked whether you want to save the base (key 1), the files (key 2) or the whole ledger (key 3). Pressing any other key aborts the routine, returning you to the menu.

In the majority of cases you will want to save the whole ledger on tape (key 3), but the base and file save facilities can be useful if you want to save only the files and incorporate a new base in the ledger with the 'Load' routine 'Merge' facility, or if you want to use the 'New Base' facility to 'Chain' a number of ledgers together (see below). Whatever option you choose, the program recognises the importance of your ledger files and saves two copies — just in case a fault in the tape causes a loading problem. The message 'Start Tape' will appear on the screen, followed by a short delay (to allow the tape leader to go past the record head), then the computer will announce it is saving 'Copy 1', 'Copy 2', then return to the menu.

The ledger 'header' (the important ledger information) is saved at 300 baud, then the base and files are saved at 1200 baud. This is not a job for bargain-basement tapes. I use TDK 120 us bias cassettes — anything less is just begging for an unloadable file that you'll have to type all over ag Database 'Save' routines tend to bring out the worst in Microbee casse... interfaces; readers whose Bees are 32 Kbyte IC models or earlier should consider making the tape I/O modifications suggested in the *Microbee Hacker's Handbook* (available through *Your Computer*), and those with the Telcom v1.0 or 1.1 communications ROM ought to either update to a later version or have the permanent RTC link disconnected. Otherwise, loading at 1200 baud becomes a game of chance!

Load

The load facility is designed for maximum flexibility in manipulating your ledgers. Not only can you load a full ledger (key 3) or a base or files alone (keys 1 or 2), but if there is already a ledger in memory and you choose to load only the files or base of your tape ledger, that section will replace the corresponding section of the ledger already in memory. This allows you to create a new base if the old one was deficient (such as if you decide an extra category ought to have been included in the payments section).

A 'Merge' facility (key 4) is also included, allowing you to add the files in your tape ledger to the end of the ledger (if any) already in memory. If this exceeds the ledger capacity (320 files), the computer announces "Merge - Part only", and you'll have to use the 'Display' function to check how many of the tape files were loaded before the ledger reached its capacity.

Once you've selected the load option you want to use, the program requests you to 'Start Tape', and this message remains on the screen until a valid tape header is encountered. At this stage the file title will be displayed with some additional information (such as whether the t consists of a base or files only or whether the files are unsorted — below). If the tape file is incompatible with your load command (such as if you commanded the computer to load a base and the tape dump is files only), the routine will abort leaving any existing ledger intact. Please note, limits on program length meant that no load indicator or error checking could be included, so if your ledger hasn't loaded after about two minutes then press <RESET> and try again.

Sort

This routine arranges the files in your ledger in chronological order according to the file dates. As LedgerMaster is written in BASIC the Sort routine is quite slow (about five minutes to sort 320 files), so you will be asked whether you are 'Sure?'. Press key 'N' to abort or key 'Y' to continue.

Your files don't have to be sorted before they're saved on tape or printed, but your final printout will be much easier to follow if a sort is first carried out. A counter will appear on the screen as the routine is in progress (just to reassure you your computer hasn't died of a stroke!).

Delete

This function is useful if you want to get rid of an incorrectly typed file or a slab of files in a ledger that has been partly 'merged' with an earlier ledger. Enter the number of the first file to be deleted and the file will appear in the display window. Now press key 'N' if you've picked the wrong one or key 'Y' if it's correct. You'll then be asked for the number of the last file to be deleted. If only one file is to be deleted, press the <RETURN> key, otherwise enter the appropriate file number and check that the file that

appears on the display window is the last one in the section you want to erase. Both files that appeared on the screen will be deleted, along with all files in between. Invalid file numbers will be rejected.

Print

This is a powerful function with a range of options. You'll first be asked whether you want to search for any particular string, allowing you to search through the date, item, receipt number and/or money columns of the ledger for all occurrences of a particular set of characters. So if you entered 'B/card <RETURN>' in response to the 'Search' prompt, the only ledgers printed would be those containing this expression.

If you're searching the money column for all entries of a particular amount, please remember trailing zeros in the cents column are omitted. A hash mark (#) followed by a file number entered in response to the search request will cause printing to commence at that file number. To print all files in the ledger simply press the <RETURN> key.

The program will now ask whether you want the printout directed to the screen or to a printer (keys 'S' or 'P'). If you select the printer option a sub-menu will appear, asking you to select the printer type (press an invalid key if you chose the printer option by mistake). Once you've made your selection you'll be asked to press a key to continue. Check your paper is properly aligned in the printer, press a key and away it will go.

The ledger printout consists of three sections (see Figure 1). First are the categories 'carried forward' (if any), followed by the ledgers themselves and finally the balance sheet. Base categories with a nil balance are omitted. If the 'search' mode is used then only the ledgers are printed, as otherwise the balance sheet totals would be misleading.

When printing on the screen the ledger categories are shown simply by their code letter (to accommodate the 64-character-width screen), but these are printed in full if a printer is used. The printer format also includes a page heading showing the ledger name, column headings, page number and (if the ledger has been sorted) the dates it covers.

New Base

This option assists in 'chaining' ledgers by taking the base categories of an existing ledger, adding to each the total of all applicable files, then using the files themselves. This new base now contains 'carried forward' totals that are used in the new ledger, so its balance sheet reflects the total of the entries in both.

When the option is selected you'll first be asked if you're 'Sure?', as accidental selection of this routine would have disastrous results. You'll then be asked to enter the name of the new ledger, following which the necessary operations will be carried out and the program will return to the menu.

A Final Balance

No doubt some readers will have the utter horrors at typing in a program of this length. If you want to save yourself the trouble, forward a cheque or money order for $15, payable to Dreamcards, 8 Highland Court, North Eltham 3095 (mail order only), and I'll send you a tape.

Anyone wanting a more sophisticated tape-based accounting system might like to consider my Multibank program ($34.95 plus $1.50 postage and packing, for cassette and manual). This generates cassette files containing up to eight ledgers, and has heaps of frills usually only found on expensive disk-based systems (such as automatic allowance for periodical debits).

Coming Soon

Next month's issue of *Your Computer* will tell you how LedgerMaster works and how to convert it to run with other BASICs. □

LedgerMaster

```
00001 POKE 140,1  REM Disable <BREAK> Key
00002 GOSUB 333  REM Initialise Storage arrays
00003 IN#0: OUT#0  REM VDU On
00004 N=8: IF B0$(0)="" THEN LET N=2 ELSE IF C=1 THEN LET N=4
00005 K0$=KEY$: K4$="": I=0: Q=0: IF C<3 AND N=8 THEN LET N=5
00006 GOSUB 298: INVERSE: CURS 28,4: PRINT "^MENU^": NORMAL
00007 PRINT \ SPC(14) "Create^.....^1";
00008 PRINT SPC(4) "Load^.....^2": IF N=2 THEN 16
00009 PRINT SPC(14) "Enter^.....^3";
00010 PRINT SPC(4) "Save^.....^4": IF N=4 THEN 16
00011 PRINT SPC(14) "Delete^.....^5";: IF N=5 THEN 16
00012 PRINT SPC(4) "Sort^.....^6"
00013 PRINT SPC(14) "Print^.....^7";
00014 PRINT SPC(4) "New^Base^...^8"
00015 IF C=V+1 THEN PRINT \ SPC(21) "***^LEDGER^FULL^***"
00016 K2$="Select^Menu^Option": GOSUB 226
00017 X=INT(VAL(K0$)): IF X=0 OR X>N THEN 16
00018 PLAY 22,1: ON X GOTO 65,33,83,120,138,22,150,216
00019 REM
00020 REM ------------------ Sort Routine ------------------
00021 REM
00022 GOSUB 232: IF K1$="N" OR S=1 THEN 3
00023 X=0: GOSUB 317  REM Set-up Date array
00024 FOR X=1 TO C-1: GOSUB 292: A1(X)=F4: NEXT X
00025 FOR X=1 TO C-2: GOSUB 317: FOR M=X+1 TO C-1  REM Sort
00026 IF A1(M)=>A1(X) THEN 29
00027 K0$=A0$(M): A0$(M)=A0$(X): A0$(X)=K0$
00028 F1=A1(M): A1(M)=A1(X): A1(X)=F1
00029 NEXT M: NEXT X: S=1: GOTO 79
00030 REM
00031 REM ----------------- "Load" Routine -----------------
00032 REM
00033 N=3: IF C>1 AND C<V+1 THEN LET N=4
00034 GOSUB 324: IF O=0 THEN 3
00035 OUT#0 OFF: IN#2  REM VDU Off, 300 baud Cass. in
00036 INPUT K1$, D, H, X, Y, Z, E: K1$=K1$(;7)
00037 IN#0: OUT#0  REM VDU On again
00038 IF O=3 AND H<3 THEN LET O=0
00039 IF O=1 AND H=2 THEN LET O=0
00040 IF O>1 AND H=1 THEN LET O=0
00041 K0$="": J=1: L=D-1: IF O<4 THEN 44
00042 J=C: L=L+C: C=L: S=0: K0$="^-^Merge"
00043 IF C>V+1 THEN LET C=V+1: K0$=K0$+"^(Part^only)"
00044 IF O=2 OR O=3 THEN LET C=D: S=Z
00045 IF O=1 OR O=3 THEN LET B0$(0)=K1$: P=X: R=Y: T=E
00046 K1$=K1$+K0$: CURS 0,12: PRINT [A64^32]
00047 CURS (64-LEN(K1$))/2,12: PRINT K1$: CURS 26,13
00048 IF H=1 OR O=1 OR C=1 THEN PRINT "^^(Base)": GOTO 51
00049 IF (H=2 OR O=2) AND C>1 THEN PRINT "^^(File)": GOTO 51
00050 IF S=0 AND C>1 AND O>1 THEN PRINT "(Unsorted)"
00051 IF O=0 THEN 79 ELSE LET N=X: IF Y>X THEN LET N=Y
00052 CURS 0: OUT#0 OFF: IN#3  REM VDU Off, 1200 baud Cass. in
00053 FOR X=1 TO N  REM Load Base
00054 IF (O=2 OR O=4) AND H=3 THEN INPUT K0$, K0$, F1, F1
00055 IF O=1 OR O=3 THEN INPUT B0$(X), B1$(X), B2(X), B3(X)
00056 NEXT X: IF O=1 THEN 58 ELSE FOR X=J TO C STEP 3
00057 INPUT A0$(X), A0$(X+1), A0$(X+2): NEXT X
00058 IF C<V+1 AND O>1 THEN GOSUB 336
00059 Y=P+1: IF P<20 THEN GOSUB 337
00060 Y=R+1: IF R<20 THEN GOSUB 338
00061 GOTO 79
00062 REM
00063 REM ---------- ------ "Create" Routine ----------------
00064 REM
00065 GOSUB 331: IF K1$="N" THEN 3
00066 G=24: K2$="File^Title": GOSUB 298: GOSUB 250: B0$(0)=K1$: I=1
00067 GOSUB 304: P=P+1: GOSUB 249: B0$(P)=K1$: K3$=K1$
00068 IF K3$="" AND P=1 THEN LET B0$(1)="Payment": GOTO 70
00069 IF K3$="" THEN LET P=P-1: GOTO 71
00070 GOSUB 237: B2(P)=F3: IF P<20 AND K3$<>"" THEN 67
00071 PLAY 22,1
00072 GOSUB 307: R=R+1: GOSUB 249: B1$(R)=K1$: K3$=K1$
00073 IF K3$="" AND R=1 THEN LET B1$(1)="Receipt": GOTO 75
00074 IF K3$="" THEN LET R=R-1: GOTO 76
00075 GOSUB 237: B3(R)=F3: IF R<20 AND K3$<>"" THEN 72
00076 PLAY 22,1: T=0: FOR X=1 TO 20: IF B2(X)>0 THEN LET T=T+1
00077 IF B3(X)>0 THEN LET T=T+1
00078 NEXT X: C=1: K3$="": GOTO 3
00079 CURS 0: PLAY 22,1; 0,5: GOTO 3
00080 REM
00081 REM ----------------- "Enter" Routine -----------------
00082 REM
00083 IF C>V THEN 3
00084 S=0: A0$(C)="": M=C: GOSUB 268: I=C-1: G=2
00085 IF C>1 THEN LET X=C-1: GOSUB 292
```

▷

LedgerMaster

```
00086 K2$="Day": GOSUB 250: IF K1$="" AND F1>0 THEN 92
00087 F1=VAL(K1$): IF F1=0 OR F1>31 THEN 86
00088 K2$="Month": GOSUB 250: IF K1$="" AND F2>0 THEN 92
00089 F2=VAL(K1$): IF F2=0 OR F2>12 THEN 88
00090 K2$="Year": GOSUB 250: IF K1$="" AND F3>0 THEN 92
00091 F3=VAL(K1$): IF F3=0 THEN 90
00092 RESTORE: FOR X=1 TO INT(F2): READ Z: NEXT X
00093 DATA^31,28,31,30,31,30,31,31,30,31,30,31
00094 IF FRACT(F3/4)=0 AND F2=2 THEN LET Z=29
00095 IF INT(F1)<=Z THEN 97
00096 CURS 1,16: PRINT "***^DATE^ERROR^***";: PLAY 10,12: GOTO 86
00097 K0$=STR$(INT(F1)): K1$=K0$(;2)+"/"
00098 K0$=STR$(INT(F2)): K1$=K1$+K0$(;2)+"/"
00099 K0$=STR$(INT(F3)): IF F3<10 THEN LET K1$=K1$+"0"
00100 K1$=K1$+K0$(;2): GOSUB 267: K2$="From/To": G=16: I=1
00101 GOSUB 250: IF K1$="" THEN LET K1$="Paid"
00102 GOSUB 267: K2$="Cheque/Receipt^No.": G=7: GOSUB 250
00103 IF X=2 THEN LET K1$="B/card" ELSE IF X=3 THEN LET K1$="Cash"
00104 GOSUB 267: K2$="Receipt^(R)^or^Payment^(P)"
00105 GOSUB 226: IF K1$<>"R" AND K1$<>"P" THEN 105
00106 GOSUB 285: GOSUB 298
00107 IF K1$<>"P" THEN 109
00108 M=P: IF M=1 THEN LET K1$="A": GOTO 111 ELSE GOSUB 304: GOTO 110
00109 M=R: IF M=1 THEN LET K1$="A": GOTO 111 ELSE GOSUB 307
00110 K2$="Category": GOSUB 226: X=X-64: IF X<1 OR X>M THEN 110
00111 GOSUB 267
00112 G=9: I=0: K2$="Amount": GOSUB 250: GOSUB 242: IF X=1 THEN 112
00113 GOSUB 267: K2$="Next=<CR>^^Menu=M^^Re-do=R^":I=1
00114 GOSUB 226: IF K1$="R" THEN 84
00115 IF K1$<>"M" AND X<>128 THEN 114
00116 C=C+1: IF X=128 THEN 83 ELSE 3
00117 REM
00118 REM ----------------- "Save" Routine -----------------
00119 REM
00120 N=3: IF C=1 THEN LET N=1
00121 GOSUB 324: IF H=0 THEN 3
00122 N=P: IF R>P THEN LET N=R
00123 FOR Q=1 TO 2: PLAY 0,60: K0$=B0$(0)
00124 CURS 0,12: PRINT [A28^32]; "Copy"; Q; [A28^32]: CURS 0
00125 OUT#2  REM 300 Baud Cassette out (VDU remains on)
00126 PRINT "******"; K0$; ","; C; ","; H; ","; P; ","; R; ","; S; ","; T
00127 PLAY 0,1: OUT#3  REM Delay / 1200 Baud Cassette out
00128 IF H=2 THEN 130 ELSE FOR X=1 TO N  REM Save BASE
00129 PRINT B0$(X); ","; B1$(X); ","; B2(X); ","; B3(X): NEXT X
00130 PLAY 0,1: IF C<2 OR H=1 THEN 133
00131 FOR X=1 TO C STEP 3  REM Save FILES
00132 PRINT A0$(X); ","; A0$(X+1); ","; A0$(X+2): NEXT X
00133 OUT#0  REM De-select Cassette output
00134 NEXT Q: GOTO 79
00135 REM
00136 REM ---------------- "Delete" Routine ---- ----------
00137 REM
00138 G=3: K3$="^file^to^be^deleted": K2$="First"+K3$
00139 GOSUB 250: O=INT(VAL(K1$)): IF O<1 OR O=>C THEN 138
00140 M=O: GOSUB 268: GOSUB 232: IF K1$="N" THEN 3
00141 G=3: K2$="Last"+K3$+"^-^or^<CR>": I=1: GOSUB 250
00142 L=INT(VAL(K1$)): IF K1$="" THEN LET L=O
00143 I=O: IF L<O OR L=>C THEN 141 ELSE IF L=O THEN 145
00144 M=L: GOSUB 268: GOSUB 232: IF K1$="N" THEN 141
00145 K3$="": Z=L-O+1: X=O: GOSUB 317: FOR X=L+1 TO C
00146 A0$(O)=A0$(X): O=O+1: NEXT X: C=C-Z: GOSUB 336: GOTO 79
00147 REM
00148 REM ---------- ------ "Print" Routine ----  ----------
00149 REM
00150 G=20: I=1: H=1: K2$="String^search": GOSUB 250: K4$=K1$
00151 IF ASC(K4$)=35 THEN LET H=INT(VAL(K4$(;2))): K4$="#"
00152 IF H=0 OR H>C-2 THEN 150
00153 K2$="Screen^(S)^or^Printer^(P)": GOSUB 226
00154 L=12: IF K1$="S" THEN 163 ELSE IF K1$<>"P" THEN 3
00155 GOSUB 298: K2$="PRINTER^Type:": PRINT \\ SPC(15): K2$
00156 PRINT \ SPC(20) "Parallel^^^^^^-^^Key^1"
00157 PRINT SPC(20) "Serial^300bd^^-^^Key^2"
00158 PRINT SPC(20) "Serial^1200bd^^-^^Key^3"
00159 PRINT SPC(20) "Abort^-^^any^other^Key"
00160 GOSUB 226: X=X-48: IF X<0 OR X>4 THEN 3
00161 L=57: Q=X: IF X>1 THEN LET Q=X+2
00162 GOSUB 225: OUT#0: OUT#0 OFF: OUT#Q ON: PRINT
00163 FOR X=1 TO 20: B4(X)=B2(X): B5(X)=B3(X): NEXT X
00164 D=0: F3=0: F4=1: O=T: U=0: W=0: E=C+P+R+T+4
00165 IF K4$<>"" THEN LET E=C: O=1
00166 G=1: K0$=KEY$: IF K0$="" THEN 169 ELSE PLAY 22,2
00167 IF K0$="A" OR K0$="a" THEN 210  REM Detect <A> key
00168 K1$=KEY$: IF K1$="" THEN 168  REM Hold if any other key
00169 IF D>0 THEN 178 ELSE IF Q>0 THEN 171  REM Page Heading
00170 FOR X=3 TO 14: CURS 1,X: PRINT [A64^32]: NEXT X: CURS 1,3: GOTO 178
00171 D=2: PRINT "LEDGER:^"; B0$(0); SPC(26-LEN(B0$(0)));
00172 K2$="^to^": IF S=0 THEN 175  REM Print dates if SORTED
00173 K0$=A0$(1): GOSUB 286: K2$=K1$+K2$
00174 K0$=A0$(C-1): GOSUB 286: K2$=K2$+K1$: PRINT K2$;
00175 PRINT SPC(34-LEN(K2$)); "Page^"; F4: F4=F4+1
00176 PRINT "No.^^^^^DATE^^^^TRANSACTION"; SPC(17);
00177 PRINT "CLASS"; SPC(16); "AMOUNT^($)": PRINT [A79^45]
00178 IF O=0 OR K4$<>"" THEN 188  REM Print "Carried Forward"
00179 U=U+1: IF U>P THEN 182
00180 F1=B2(U): IF F1=0 THEN 179
00181 K0$=B0$(U): X=LEN(K0$): Z=0: GOTO 184
00182 W=W+1: F1=B3(W): IF F1=0 THEN 182
00183 K0$=B1$(W): X=LEN(K0$): Z=11
00184 IF O=T THEN PRINT "^CARRIED^FORWARD:^^^"; ELSE PRINT SPC(20);
00185 IF Q>0 THEN PRINT SPC(22); K0$; SPC(27-X-Z);
00186 IF Q=0 THEN PRINT K0$; SPC(33-X-Z);
00187 O=O-1: PRINT [F10.2^F1]: GOTO 205  REM Print amount carried
00188 IF K4$="" OR K4$="#" OR O<>1 THEN 190
00189 PRINT \ SPC(10); "SEARCH;^"; K4$ \: O=0: G=3: GOTO 205
00190 M=H: IF K4$="" THEN LET M=H-T
00191 IF M<1 OR M>=C THEN 194 ELSE GOSUB 269  REM Print FILE
00192 IF M=0 THEN LET B5(N)=B5(N)+F1 ELSE LET B4(N)=B4(N)+F1
00193 GOTO 205  REM Add balance in file to BASE category
00194 IF K4$<>"" THEN 205
00195 M=M-C: IF M=0 THEN PRINT: GOTO 205  REM FILE End
00196 IF M>P THEN 199 ELSE IF B4(M)=0 THEN 209  REM Payments
00197 PRINT SPC(20); B0$(M); SPC(22-LEN(B0$(M))); [F10.2^B4(M)]
00198 F3=F3-B4(M): GOTO 205
00199 M=M-P: IF M>R THEN 202 ELSE IF B5(M)=0 THEN 209  REM Receipts
00200 PRINT SPC(20); B1$(M); SPC(22-LEN(B1$(M))); [F10.2^B5(M)]
00201 F3=F3+B5(M): GOTO 205
00202 X=67: IF Q=0 THEN LET X=51  REM Print Total at end
00203 IF H=E OR H=E-2 THEN PRINT SPC(X+2); [A10^45]: G=1
00204 IF H=E-1 THEN PRINT SPC(X); [F11.2^F3]: G=1
00205 D=D+G: IF D<L THEN 209  REM Lines per page counter
00206 REM Delete next line if not using fan-fold paper
00207 IF Q>0 THEN PRINT \\\\\\\  REM Perf. gap = 7 lines
00208 D=0: IF Q>0 THEN 209 ELSE GOSUB 225: IF K0$="A" THEN 212
00209 H=H+1: IF H<=E THEN 166  REM Detect end of printing
00210 IF Q>0 THEN OUT#Q OFF: GOTO 212
00211 IF K0$<>"A" AND K0$<>"a" THEN GOSUB 225
00212 Q=0: GOTO 3
00213 REM
00214 REM ---------- "New Base" Routine ----------
00215 REM
00216 GOSUB 232: IF K1$="N" THEN 3
00217 I=1: G=24: K2$="New^Title": GOSUB 250: IF K1$<>"" THEN LET B0$(0)=K1$
00218 FOR Y=1 TO C-1: K0$=A0$(Y): X=SEARCH(K0$,"|",3): GOSUB 287
00219 O=ASC(K0$): GOSUB 286: Z=ASC(K0$)-64: GOSUB 286: F1=VAL(K0$)
00220 IF O=80 THEN LET B2(Z)=B2(Z)+F1 ELSE LET B3(Z)=B3(Z)+F1
00221 X=Y: GOSUB 317: NEXT Y: C=1: GOSUB 336: GOTO 76
00222 REM
00223 REM ---------- "Key to Continue" Sub. ----------
00224 REM
00225 K2$="KEY^to^Continue"
00226 G=1: GOSUB 250: X=ASC(K1$)
00227 IF X>96 AND X<123 THEN LET X=X-32: K1$=CHR$(X)
00228 RETURN
00229 REM
00230 REM ---------- "Sure (Y/N)" Sub. ----------
00231 REM
00232 K2$="Sure^(Y/N)": GOSUB 226: IF K1$<>"N" AND K1$<>"Y" THEN 232
00233 RETURN
00234 REM
00235 REM ---------- "Previous Balance" Input Sub. ----------
00236 REM
00237 G=9: K2$="Balance^Forward": GOSUB 250: GOSUB 242
00238 IF X=1 THEN 237 ELSE RETURN  REM Loop if invalid amount
00239 REM
00240 REM ---------- "Money Input" Sub. ----------
00241 REM
00242 X=1: F1=VAL(K1$): IF F1<=0 AND I=0 OR F1=>1000000 THEN 245
00243 X=0: F2=FRACT(F1): F3=F1-F2+FLT(INT(F2*100))/100
00244 K1$=STR$(F3): K1$=K1$(;2)  REM Round off to 2 places
00245 RETURN
00246 REM
00247 REM ---------- Main "Input" Sub. ----------
00248 REM
00249 G=12: K2$="Category"
00250 GOSUB 299: Y=LEN(K2$): CURS 1,15: PRINT [A64^45] [A63^32];
00251 X=V+1-C: IF X>V THEN LET X=V  REM Limit free files counter
00252 CURS 1,16: PRINT K2$;: CURS 52,16: PRINT INT(FRE($)); "^:"; X;
00253 CURS Y+3,16: Z=0: K1$="": FOR X=1 TO G: PRINT "*";: NEXT X
00254 CURS Y+1,16: PRINT "?^";  REM Print prompt line
00255 K0$=KEY$: X=ASC(K0$): IF X=124 OR X=128 THEN 255
```

```
00256 IF X=8 OR X=127 THEN 261 ELSE IF X>13 THEN 259
00257 IF Z>0 THEN CURS 0: RETURN  REM End if <CR>
00258 IF I>0 THEN LET K1$="": RETURN ELSE 255
00259 Z=Z+1: CURS Y+Z+2,16: PRINT KO$;  REM Print Input so far
00260 K1$=K1$+KO$: IF Z=G THEN RETURN ELSE 255  REM End of string?
00261 KO$=K1$(;1, LEN(K1$)-1): K1$=KO$  REM <DEL>ete Key
00262 IF Z>0 THEN CURS Y+Z+2,16: PRINT "*";: CURS Y+2,16: Z=Z-1
00263 GOTO 255
00264 REM
00265 REM ------------- "File entry" Print Sub. ------------
00266 REM
00267 M=C: GOSUB 285  REM "M" is record number
00268 GOSUB 298: PRINT "FILE:"
00269 IF K4$="" OR K4$="*" THEN 271  REM Implement "Search"
00270 KO$=AO$(M): IF SEARCH(KO$,K4$)=0 THEN LET G=0: RETURN
00271 KO$=STR$(M)+".": KO$=KO$(;2): X=LEN(KO$)
00272 PRINT KO$; SPC(4-X);: KO$=AO$(M): GOSUB 286  REM Date
00273 PRINT SPC(9-LEN(K1$)); K1$; "^";: IF Q>0 THEN PRINT "^";
00274 GOSUB 286: PRINT K1$; SPC(18-X);  REM Item
00275 GOSUB 286: PRINT K1$; SPC(9-X);  REM Cheque/Receipt No.
00276 GOSUB 286: M=0: IF K1$="P" THEN LET M=11  REM Rec/Pay?
00277 IF Q=0 THEN PRINT K1$; "^";  REM Don't print in full on VDU
00278 GOSUB 286: N=ASC(K1$)-64: K2$=BO$(N): IF M=0 THEN LET K2$=B1$(N)
00279 IF Q>0 THEN LET X=LEN(K2$): PRINT K2$; SPC(16-X); ELSE PRINT K1$;
00280 GOSUB 286: F1=VAL(K1$): IF F1>0 THEN PRINT SPC(M); [F10.2^F1]
00281 RETURN
00282 REM
00283 REM ------------ Extract file section Sub. -----------
00284 REM
00285 AO$(C)=AO$(C)+K1$+"|": RETURN  REM "|" delimits section
00286 K1$="": X=SEARCH(KO$,"|"): IF X=0 THEN RETURN
00287 K1$=KO$(;1,X-1): KO$=KO$(;X+1): IF Q>0 THEN LET X=X-1
00288 RETURN
00289 REM
00290 REM ------------ Extract Date Variables Sub. ----------
00291 REM
00292 K1$=AO$(X): F1=VAL(K1$): Y=SEARCH(K1$,"/")+1: K1$=K1$(;Y)
00293 F2=VAL(K1$): Y=SEARCH(K1$,"/")+1: K1$=K1$(;Y)
00294 F3=VAL(K1$): F4=100*F3 + F2 + F1/100: RETURN
00295 REM
00296 REM ------------ Print Screen heading Sub. -----------
00297 REM
00298 CLS
00299 CURS 17,1: INVERSE: PRINT "^*^Dreamcards^LedgerMaster^*^"
00300 NORMAL: PRINT [A64^45];: RETURN
00301 REM
00302 REM ----------------- Print BASE Sub. ----------------
00303 REM
00304 GOSUB 298: PRINT "PAYMENT^CATEGORIES:^^("; B0$(0); ")": K2$="Category"
00305 IF P=0 THEN RETURN  REM Payments
00306 FOR X=1 TO P: KO$=BO$(X): F1=B2(X): GOSUB 310: NEXT X: RETURN
00307 GOSUB 298: PRINT "RECEIPTS^CATEGORIES:^^("; BO$(0); ")"
00308 IF R=0 THEN RETURN  REM Receipts
00309 FOR X=1 TO R: KO$=B1$(X): F1=B3(X): GO$UB 310: NEXT X: RETURN
00310 J=X+3: K=3: IF X>10 THEN LET J=J-10: K=35
00311 CURS K,J: PRINT CHR$(X+64); ".^"; KO$; SPC(13-LEN(KO$));
00312 IF F1>0 THEN PRINT [F10.2^F1]  REM Print any balance
00313 RETURN
00314 REM
00315 REM -------------- "Counter" Print Sub. --------------
00316 REM
00317 IF X>1 THEN 319
00318 CURS 1,12: PRINT [A26^32] "*^WAIT^*" [A92^32]
00319 IF X>0 THEN CURS 28,13: PRINT [I4^X];
00320 CURS 0: RETURN
00321 REM
00322 REM -------------- Tape I/O Prompt Sub. --------------
00323 REM
00324 K2$="Base^(1)": IF N>2 THEN LET K2$=K2$+",^Files^(2),^All^(3)"
00325 IF N>3 THEN LET K2$=K2$+",^Merge^(4)"
00326 GOSUB 226: H=INT(VAL(K1$)): IF H>N THEN LET H=0
00327 O=H: CURS 26,12: PRINT "Start^tape": CURS 0: RETURN
00328 REM
00329 REM ------------- Initialise Arrays Sub. -------------
00330 REM
00331 IF C=0 THEN 333  REM Print prompt if existing file
00332 GOSUB 232: IF K1$="N" THEN RETURN
00333 CLEAR: V=320: STRS(V*51+1400)  REM V = Max. No. of files
00334 DIM AO(V+2), A1(V+2), BO(20), B1(20), B2(20)
00335 DIM B3(20), B4(20), B5(20): GOSUB 337: GOSUB 338
00336 FOR X=C TO V+2: AO$(X)="": NEXT X: K1$="": RETURN
00337 FOR X=Y TO 20: BO$(X)="": B2(X)=0: NEXT X: RETURN
00338 FOR X=Y TO 20: B1$(X)="": B3(X)=0: NEXT X: RETURN
```

CB80/86 Standard I/O Library

The lengthy* CBASIC *Compiler* (CB80 *and* CB86) *listings here may seem a bit much for a single program — and they are. We're including our in-house function library, and its support files, both to make your own programming easier and to ensure you have the library available in future when we publish other programs which use it.

SEVERAL OF THE FUNCTIONS listed in the following pages are based on a set of PL/I routines written by John Skaller for our subscriptions system. They are in many ways an effective replacement for Digital Research's Display Manager — John had to write them because our DM86 hadn't arrived by the time we had to have our system running.

The dBase file fixer FIXDBF.BAS uses several of the functions in the library, but certainly not all of them — you can be selective about those you type in if you wish. However, we recommend you take the complete library, and compile and LIB it for easy use in the future.

The files which make up the function library are these:

STDIO.BAS: The function definitions themselves (we'll explain them one-by-one later). These can be used as is by %INCLUDEing them in your source file, or defined as PUBLIC for library creation and an appropriate set of EXTERNAL declarations created.

STDIO.DCL: These are declarations of the global variables used by the functions (and by the programs themselves). All are declared COMMON, to allow for the separate library approach.

STDIO.ASS: This file assigns values to the variables declared in the DCL file, and is normally %INCLUDEd at the start of your source file.

STDIO.TRM: Most of the functions use direct screen addressing and other terminal-specific control codes. The default values in the library are for terminals in the Televideo/ADM style, which may not suit you. Alternative codes can be put in this file, and a call to the GET.PARMS function will install these values in your program.

Now, let's look at the functions themselves:

Unbug(v$) is useful for debugging a program — it is passed a message which is printed to the status line of the TVI950 terminal.
Conout(v$), coninp$, and conprt(msg$) are simple 'print a character', 'get a character' and 'print a string' routines used by the I/O functions. We have versions which use BDOS calls, but suggest you use these simplified ones for portability.
Terminit, highlight, lowlight, reverse, unrevrs, clrscr, clreol and bell are self-explanatory.
Gotoxy(x%,y%) allows direct screen addressing. It is fed x and y co-ordinates, which should be 1 to 24 and 1 to 80 respectively. If it is given a co-ordinate of 0, it assumes the last-addressed position. (There is no provision for terminals which send column before row.)
Displ(x%,y%,v$) displays the string v$ at the co-ordinates specified, while **displyl(x%,y%,v$)** does the same and clears to end-of-line.
Fmtdate$(da$) turns a date string in the form 850501 around to 01/05/85.
Fmtnum$(de,le%) takes a number and turns it into a string of a specified length to allow easier screen editing.
Fmtdol$(de,le%) does the same, but in decimal format (that is, it shows 1000 as 10.00).
Valid$(q$,type%) is used by the input functions to validate the operator's input, which can be forced to alpha, numeric, upper case or only alpha.
Vinput$(x%,y%,n%,type%) is a string input routine. At the x-y position specified it puts n% dots (the length of the string to be input) and accepts only type% input. Backspacing will delete the previous character and replace it with a dot. Valid type% values are alpha, numeric, upper and onlyalpha (assigned in STDIO.ASS). **Linput$(x%,y%,n%)** is a simpler way to use vinput — it assumes a type of alpha.
Vupd$(x%,y%,deflt$,type%) is a string update routine which works in much the same way as Vinput, but with an existing default. **Lupd$(x%,y%,deflt$)**, like Linput, assumes a type of alpha.
Syserr(x%,y%,v$) prints an error message v$ at the specified co-ordinates and waits for a key to be hit before continuing.
Yesno%(x%,y%,v$,d%) puts the question v$ at the specified position, follows it with 'Y/N', and positions the cursor on the default answer (d%). The default can be yes, no, or none — if it is none the cursor is positioned on the slash and an answer of Y or N is forced.
Verify%(tmpstr$,patrn$) compares a string to a given pattern to confirm that only legal characters (that is, those in the pattern) are includ-
Vdate%(vddmmyy$) verifies a date passed as ddmmyy, **Revdate$(mmyy$)** reverses a date from ddmmyy to yymmdd or vice versa.
Ninput(x%,y%,fln%,minval,maxval,defval) is like Vinput, but for numbers, and allows you to specify length (fln%), minimum, maximum and default values.
Nupd(x%,y%,fln%,minval,maxval,defval,deflt) provides numeric update, like Lupd, but allows for an alternative default. Note that in Ninput and Nupd all numbers are converted to strings for the sake of convenient screen handling, and converted back again later.
Doscan%(patrn$,base$) checks how often the patrn$ occurs in the string base$. It is a support routine for Isvalid$. Both functions are adapted from a Digital Research sample program.
Isvalid$(fname$) checks the validity of a filename, returning a null string if it is incorrect.
Chrconv$(v$) is a support routine for Get.parms. It converts decimal ASCII values to the appropriate characters for screen control sequences.
Get.parms allows you to change screen control strings to suit different terminals. It looks on the disk for a file called STDIO.TRM and, if it finds it, replaces the values in memory with the new values.
Trim$(v$) trims trailing blanks from the string v$.
Delay(length) provides a delay of approximately *length* seconds. This depends on dlyfactor, set in STDIO.ASS, which will vary from machine to machine. The default of 250 works on a 6 MHz 8085.
Err.handler is an error handler which simply prints the error code and exits. It assumes the error is fatal, and is provided mainly for use during the development phase of a program. □

```
\=========================================UNBUG
def unbug(v$)
    rem print to status line of kokusai
    print escape+"g"+escape+"f"+v$+chr$(13);
    ko%=inkey
fend

\=========================================CONOUT
def conout(v$)
    print v$;
fend

   =======================================CONINP$
def coninp$
    v$=chr$(inkey)
    coninp$=v$
fend

\=========================================CONPRT
def conprt(msg$)
    for qi%=1 to len(msg$)
        call conout(mid$(msg$, qi%, 1))
    next qi%
fend

\=======================================TERMINIT
def terminit
    call conprt(trmini)
fend

\=======================================HIGHLIGHT
def highlight
    call conprt(fullintensity)
fend

\=======================================LOWLIGHT
def lowlight
    call conprt(halfintensity)
fend

\=========================================REVERSE
def reverse
    call conprt(revideo)
  nd

\=========================================UNREVRS
def unrevrs
    call conprt(normvideo)
fend

\=========================================BELL
def bell
    call conprt(bellchar)
fend

\=========================================GOTOXY
def gotoxy(x%,y%)
    rem screen 1-80 x 1-24 - 0 gives prev. posn
    if x%<>0 then crow=x%
    if y%>0                          \
       then ccol=y%                  \
    else                             \
        if y% < 0                    \
           then ccol=ccol-y%
    call conprt(xyleadin+chr$
     (crow+31)+chr$(ccol+31))
fend

\=========================================DISPL
def displ(x%,y%,v$)
    call gotoxy(x%,y%)
    for qi%=1 to len(v$)
        call conout(mid$(v$,qi%,1))
    next qi%
    ccol=ccol+len(v$)
fend

\=========================================CLREOL
def clreol
    call conprt(eraeol)
fend

\=========================================DISPLYL
def displyl(x%,y%,v$)
    call displ(x%,y%,v$)
    call clreol
fend

\=========================================CLRSCR
def clrscr
    call conprt(cls)
    ccol=1
    crow=1
fend

\=========================================FMTDATE$
def fmtdate$(da$)
    fmtdate$=mid$(da$,5,2)+"/"+     \
             mid$(da$,3,2)+"/"+     \
             mid$(da$,1,2)
fend

\=========================================FMTNUM$
def fmtnum$(de,le%)
    tmp$="                "+str$(de)
    fmtnum$=right$(tmp$,le%)
fend

\=========================================FMTDOL$
def fmtdol$(de,le%)
    dd=de/100.00
    tmp$="                "+str$(dd)
    if match(".",tmp$,1)=0 then              \
        tmp$=tmp$+".00"
    if mid$(tmp$,len(tmp$)-1,1)="." then     \
        tmp$=tmp$+"0"
    fmtdol$=right$(tmp$,le%)
fend

\=========================================VALID$

def valid$(q$,type%)
    valid$=""
    if type%=alpha then                              \
        if (asc(q$)>31 and asc(q$)<127) then         \
            valid$=q$
    if type%=numeric then                            \
        if match(q$,"-.+0123456789",1)>0 then        \
            valid$=q$
    if type%=upper then                              \
        if match(q$,                                 \
          "ABCDEFGHIJKLMNOPQRSTUVWXYZ"+              \
          " 0123456789[]<>|~`=+_()!@#$%^*-/.&,'"\
          +chr$(34),1)>0 then                        \
          valid$=q$                                  \
        else                                         \
          if match(q$,                               \
             "abcdefghijklmnopqrstuvwxyz",1)>0       \
                then                                 \
                q$=chr$(asc(q$)-32) : valid$=q$
    if type%=onlyalpha then                          \
      if match(q$,                                   \
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789."\
         ,1)>0 then                                  \
         valid$=q$
fend
```

```
\=========================================VINPUT$
def vinput$(x%,y%,n%,type%)
    integer l,i
    dim ch$(80)
    readline:
    call lowlight
    call gotoxy(x%,y%)
    for i=1 to n%
        call conout(".")
    next i
    call highlight
    call gotoxy (x%,y%)
    l=0
    goto readch
  wrong:
    call bell
  readch:
    q$=coninp$
    qq$=valid$(q$,type%)
    if qq$<>"" then q$=qq$ :        \
        if l < n% then              \
            l=l+1 :                 \
            ch$(l)=q$ :             \
            call conout(q$) :       \
            goto readch
    if l<n% and                             \
      match(q$,return.key+up.arrow+         \
      down.arrow+tab.key+escape,1)=0        \
      then 100
    v$=""
    for i=1 to l
        v$=v$+ch$(i)
    next i
    for i=l+1 to n%
        call conout(" ")
    next i
    call gotoxy(x%,y%)
    terminator$=q$
    vinput$=v$
    return
 100  if (q$=del.key or q$=home.key)          \
          then goto readline
      if not (q$=back.space or q$=left.arrow) \
          then goto wrong
    if l > 0 then                       \
        l=l-1 :                         \
        call lowlight :                 \
        call conout (back.space) :      \
        call conout (".") :             \
        call conout (back.space) :      \
        call highlight
    goto readch
fend

\=========================================LINPUT$
def linput$(x%,y%,n%)
    v$=vinput$(x%,y%,n%,alpha)
    linput$=v$
fend

\=========================================VUPD$
def vupd$(x%,y%,deflt$,type%)
    integer l,i
    dim ch$(80)
    n%=len(deflt$)
    readline:
    call gotoxy(x%,y%)
    call lowlight
    for i=1 to n%
        call conout(mid$(deflt$,i,1))
    next i
    call highlight
    call gotoxy (x%,y%)
    l=0
    goto readch
  wrong:
    call bell
  readch:
    q$=coninp$
    if q$=right.arrow then                  \
      if l < n% then q$=mid$(deflt$,l+1,1)  \
    else                                    \
      goto wrong
    qq$=valid$(q$,type%)
    if qq$<>"" then q$=qq$ :            \
        if l < n% then                  \
            l=l+1 :                     \
            ch$ (l)=q$ :                \
            call conout(q$) :           \
            goto readch
    if l<n% and                         \
        match(q$,return.key+up.arrow+   \
        down.arrow+tab.key+escape,1)=0  \
        then goto 100
    v$=""
    for i=1 to l
        v$=v$+ch$(i)
    next i
    for i=l+1 to n%
        call conout (" ")
    next i
    call gotoxy(x%,y%)
    terminator$=q$
    vupd$=v$
    return
 100 if (q$=del.key or q$=home.key)         \
        then goto readline
      if not (q$=back.space or q$=left.arrow) \
        then goto wrong
    if l > 0 then                           \
        l=l-1 :                             \
        call lowlight :                     \
        call conout (back.space) :          \
        call conout (mid$(deflt$,l+1,1)) :  \
        call conout (back.space) :          \
        call highlight
    goto readch
fend

\=========================================LUPD$
def lupd$(x%,y%,deflt$)
    v$=vupd$(x%,y%,deflt$,alpha)
    if len(v$)=0                    \
        then v$=deflt$              \
    else                            \
        if len(v$)<>len(deflt$)     \
            then v$=v$+right$       \
            (deflt$,len(deflt$)-len(v$))
    call highlight
    call displ(x%,y%,v$)
    lupd$=v$
fend

\=========================================SYSERR
def syserr(x%,y%,v$)
    call displ(x%,y%,v$+" (R)")
    call clreol
    ccol=ccol-2
    i$=linput$(0,0,0)
fend

\=========================================YESNO%
def yesno%(x%,y%,v$,d%)
    integer i,savcol
    call displ(x%,y%,v$)
    savcol=ccol
```

```
    call displ1(0, 0, " Y/N")
    if d%=yes then                  \
        call gotoxy(0,ccol-3)       \
    else                            \
        if d%=no then               \
            call gotoxy(0,ccol-1)   \
        else                        \
            call gotoxy(0,ccol-2)
    goto getans
  wrong:
    call bell
  getans:
    rem--------------1234-----5------
    i=match(coninp$,"nNyY"+return.key,1)
    if (i=0) or ((i=5) and (d%=none))      \
        then goto wrong
    if i=1 then goto nno
    if i=2 then goto nno
    if i=3 then goto yyes
    if i=4 then goto yyes
    rem since i=5 ...
    if d%=no then goto nno
    if d%=yes then goto yyes
    rem since d=none or other...
    goto wrong
    yyes:
      call displ(0,savcol," YES")
      yesno%=true
      return
    nno:
      call displ(0,savcol," NO ")
      yesno%=false
fend

\=========================================VERIFY%
def verify%(tmpstr$,patrn$)
    t%=0
    for qi%=1 to len(tmpstr$)
      if match(mid$(tmpstr$,qi%,1),patrn$,1)=0 \
        then t%=t%+1
    next qi%
    verify%=t%
  nd

\=========================================VDATE%
def vdate%(vddmmyy$)
    integer dd, mm, yy
    integer days(1)
    dim days(12)
    for qi%=1 to 12
        days(qi%)=31
    next qi%
    days(4)=30
    days(6)=30
    days(9)=30
    days(11)=30
    if verify%(vddmmyy$, "0123456789")<>0    \
        then vdate%=false : return
    dd=val(left$(vddmmyy$,2))
    mm=val(mid$(vddmmyy$,3,2))
    yy=val(right$(vddmmyy$,2))
    if mod (yy,4)=0                 \ leap year
        then days (2)=29            \
    else                            \
        days (2)=28
    if (dd<1 or mm<1 or mm>12 or    \
        dd>days(mm)) then           \
        vdate%=false:return
    vdate%=true
fend

\=========================================REVDATE
def revdate$(vddmmyy$)
    revdate$=                   \
    right$(vddmmyy$,2) +        \
    mid$(vddmmyy$,3,2) +        \
    left$(vddmmyy$,2)
fend

\=========================================NINPUT
def ninput(x%,y%,fln%,minval,maxval,defval)
    n%=fln%
    if n%=0 then n%=log(maxval)+1
  getin:
    v$=vinput$(x%, y%, n%, numeric)
    if len(v$)=0 then               \
        result=defval :             \
        goto printit
    result=0
    for qi%= 1 to len(v$)
        q%=match(mid$(v$, qi%, 1),\
                 "0123456789",1)
        result=result*10+q%-1
    next qi%
    if (result<minval) or (result>maxval)   \
        then goto getin
    printit:
    call displ (x%, y%, fmtnum$(result, n%))
    call gotoxy (x%, y%)
    ninput=result
fend

\=========================================NUPD
def nupd(x%,y%,fln%,minval,maxval,defval,deflt)
    n%=fln%
    if n%=0 then n%=log(maxval)+1
    vdeflt$=fmtnum$(deflt, n%)
    for qi%=1 to n%
    if mid$(vdeflt$, qi%, 1)=" " then       \
        vdeflt$=left$(vdeflt$,qi%-1)+"0"+   \
        right$(vdeflt$,len(vdeflt$)-qi%)
    next qi%
  getin:
    v$=vupd$(x%, y%, vdeflt$, numeric)
    if len(v$)=0 then                       \
        result=defval :                     \
        goto printit
    result=0
    for qi%=1 to len(v$)
        q%=match(mid$(v$,qi%,1),            \
                 "0123456789",1)
        result=result * 10+q%-1
    next qi%
    if (result<minval) or (result>maxval)   \
        then goto getin
  printit:
    call displ (x%, y%, fmtnum$(result, n%))
    call gotoxy (x%, y%)
    nupd=result
fend

\=========================================DOSCAN
def doscan%(patrn$,base$)
    count%=0
    for qi%=1 to len(base$)
      if match(patrn$,mid$(base$,qi%,1),1)>0  \
        then count%=count%+1
    next qi%
    doscan%=count%
fend

\=========================================ISVALID$
def isvalid$(fname$)
    while left$(fname$,1)=" "
        fname$=right$(fname$,len(fname$)-1)
    wend
```

```
    if fname$="" then goto badn
    for x%=1 to len(fname$)
      if (asc(mid$(fname$,x%,1)) < 33 or          \
        match(mid$(fname$,x%,1),"*,;<=>?[]",1)>0) \
        then goto badn
    next x%
    tmp%=match(":",fname$,1)
    if tmp%=1 or tmp%>2 then goto badn
    tmp%=match(".",fname$,1)
    if (tmp%<>0 and tmp%<(len(fname$)-3))         \
        then goto badn
    if doscan%(".",fname$)>1 then goto badn
    if doscan%(":",fname$)>1 then goto badn
    isvalid$=ucase$(fname$)
    return
  badn:
    isvalid$=""
fend

\=======================================CHRCONV$
def chrconv$(v$)
    rem backslash is escape char in match
    v$=left$(v$,match("\\",v$,1)-1)
    tmp$=""
    while match(",",v$,1)
      tmp$=tmp$+chr$(val(left$(v$,
      match(",",v$,1)-1)))
      v$=right$(v$,len(v$)-match(",",v$,1))
    wend
    tmp$=tmp$+chr$(val(v$))
    chrconv$=tmp$
fend

\=======================================GET.PARMS
def get.parms
    rem if screen control codes file \
        (STDIO.TRM) exists, use it
    termfile$="STDIO.TRM"
    term%=7
    tmp$="  "
    dim termch$(12)
    if size(termfile$)=0 then return
    if end # term% then err1
    open termfile$ as term%
    while left$(tmp$,2)<>"\\"
        read # term%;line tmp$
    wend
    for qi%=1 to 12
        read # term%; line termch$(qi%)
    next qi%
    cls=chrconv$(termch$(1))
    trmini=chrconv$(termch$(2))
    fullintensity=chrconv$(termch$(3))
    halfintensity=chrconv$(termch$(4))
    normvideo=chrconv$(termch$(5))
    revideo=chrconv$(termch$(6))
    up.arrow=chrconv$(termch$(7))
    down.arrow=chrconv$(termch$(8))
    left.arrow=chrconv$(termch$(9))
    right.arrow=chrconv$(termch$(10))
    eraeol=chrconv$(termch$(11))
    xyleadin=chrconv$(termch$(12))
    close term%
    return
  err1:
      close term%
      print:print "Badly constructed ";termfile$
      stop
fend

\=======================================TRIM$
def trim$(v$)
    while right$(v$,1)=" "
        v$=left$(v$,len(v$)-1)
    wend
    trim$=v$
fend

\=======================================DELAY
def delay(length)
    qi=(length*dlyfactor)
    for waitloop=1 to qi
        x=abs(10)
    next waitloop
fend

\=======================================ERR.HANDLER
def err.handler
    a$=err : i%=erl
    v$="Oops. Program error! Code="+     \
        a$+" line="+chr$(i%+48)
    call syserr(23,1,v$)
    i%=errx
    if a$="EX" then call syserr(24,1,    \
        "MP/M extended error - code "    \
        +chr$(i%+48))
    stop
fend
```

dBase

FIXDBF.BAS — The dBase User's Lifesaver

dBase can be a real pain in the neck, and not just because of its limitations — it's great to use, but when you lose a database file or its header is damaged you can be in real trouble.

Yes, it should be backed up ... but in the real world that often doesn't happen. When a file goes, you just have to grit your teeth and start typing again.

A number of programs are around which let you fix the record count in the dBase file header, but they solve only one of the problems. There are others — like the possibility of control characters being placed in the file, an end-of-file marker (control-Z) ending up in the wrong place or, worst of all, a power failure 'destroying' tens or hundreds of records because the buffers weren't written to disk and the header updated.

The last of these is the worst; if it's happened to you, take heart — FIXDBF has recovered hundreds of lost records for us at *Your Computer* in exactly the same situation in the past. And it can fix the other problems, too.

FIXDBF reads a database file a byte at a time and writes it out to a new file — either another database file or, if the header is completely garbled, it can remove the header and write out a text file which can then be appended to a new database.

It can be invoked with the two (old and new) filenames on the command line; otherwise it will prompt for the names. It then asks several questions: whether you want to produce a database file or a text file; whether you want to replace the record count (if you do, it then asks whether you want to provide the new count yourself; otherwise it 'measures' the file, and supplies its own to suit); whether you want to filter control-Zs in the body of the file; and whether you want to filter other control characters in the file. If you say yes to filtering control characters it asks whether you want them replaced automatically (with a space) or whether it should ask you each time.

It then lists the database structure (so you can tell if it is garbled) and asks whether the record size contained in the structure is correct — if not, you can nominate the record size yourself.

It's been a life-saver for us many times — hopefully it can do the same for you if you're unlucky enough to need it.

Matt Whelan

If you want to save yourself some typing, or don't have a CBASIC Compiler, you can get a compiled version of FIXDBF on single-sided CP/M 20 cm diskette or IBM PC 13 cm diskette by sending a cheque or money order for $25 to MotorWord Pty Ltd, 5 Clifton Road, Clovelly 2031. Compiled versions are available for CP/M-80, CP/M-86 and PC-DOS — please specify which operating system with your order.

```
%include stdio.dcl  rem - declare globals

integer recsize,ccount,zcount,actual.recs
integer textfile,clean.finish,dbasefile
integer scan,found,dummies,zfilter,cpmeof
integer cfilter,fixhdr,disphdr
integer last.cpmeof,askhdr
string  in.file.name,out.file.name
string  cr,nullname,crlf,vers,revision

%include stdio.ass  rem - globals assignments

cr=chr$(13)           : crlf=chr$(13)+chr$(10)
vers = "2"            : revision = "0"
cpmeof = 26           : control% = 31
maxasc%  = 127        : dbasefile = 2
infile%  = 1          : outfile% = 2
count% = 0            : recount% = 0
working%=50           : newline%=0
zfilter=false         : cfilter = false
fixhdr=false          : disphdr=false
dbheader%=521     rem dbase 521-byte header
zcount=0              : ccount=0
ask.replace=false     : askhdr=false
clean.finish=false    : textfile=false
coloffset%=15
copyrecs%=10000   rem initialise to lge no. recs
nullname=string$(10,chr$(0)) rem empty dbase fld

%include stdio.bas  rem - i/o and misc functions

on error goto errhandler
call get.parms        rem - get terminal control
                      rem - codes if they exist
call clrscr
call displ1(5,25,                                \
  "dBase Data File Fixer "+vers+"."+revision)
call displ1(6,20,                                \
  "Matt Whelan, Your Computer magazine")

\see if filenames specified on command line,
\else get file names

a$=command$
if a$="" then                                    \
    gosub getname                                \
else                                             \
    gosub breakname
rem check which operations are wanted
gosub getchoices
gosub namecheck
dim header.array%(dbheader%)

main:
    if end # infile% then finito
    if end # outfile% then ofile.error
    open in.file.name as infile%
    create out.file.name as outfile%

    gosub read.header
    if textfile then zfilter=true:               \
       cfilter=true:ask.replace=false
    if not textfile then                         \
       gosub display.header
    if askhdr then                               \
       gosub ask.header                          \
    else                                         \
       if fixhdr then                            \
          gosub fix.header
    if not textfile then                         \
       gosub write.header
    gosub copy.database
finito:
```

```
    if clean.finish                                \
       then gosub fini2                            \
    else                                           \
       gosub finished
    if not textfile then                           \
       gosub print.stats
    stop

getname:
    call displ1(crow+3,20,                         \
         "dBase file to read from? ")
    fname$=vinput$(0,0,14,alpha)
    in.file.name=isvalid$(fname$)
    call displ1(crow+1,20,                         \
         "New file to write to   ? ")
    fname$=vinput$(0,0,14,alpha)
    out.file.name=isvalid$(fname$)
    return

breakname:
    m%=match(" ",a$,1)
    if m%=0 then                                   \
        if ucase$(a$)="HELP" then                  \
            goto helpinfo                          \
        else                                       \
            goto usage
    in.file.name=isvalid$(left$(a$,m%-1))
    out.file.name=isvalid$(right$(a$,len(a$)-m%))
    return

namecheck:
    if in.file.name="" or                          \
      out.file.name="" then                        \
      goto usage
    m%=match(".",in.file.name,1)
    if m%=0 then                                   \
       in.file.name=in.file.name+".DBF"
    if size(in.file.name)<1 then                   \
       call syserr(23,1,                           \
      "Can't find file "+in.file.name)             \
       : print : goto usage
    m%=match(".",out.file.name,1)
    if m%=0 then                                   \
        if not textfile then                       \
           out.file.name=                          \
           out.file.name+".DBF"                    \
        else                                       \
           out.file.name=                          \
           out.file.name+".TXT"
    if size(out.file.name)>0 then                  \
        print:print:                               \
        if not yesno%(23,1,                        \
          "Output file exists...overwrite?"        \
          ,yes) then stop                          \
        else                                       \
           call displ1(23,1," ")
    if in.file.name=out.file.name then             \
        call syserr(23,1,                          \
        "Sorry, you can't write the"+              \
        " file to itself"):                        \
        stop
    return

getchoices:
    if not yesno%(crow+3,coloffset%,               \
        "Copy database(Y), or produce "+           \
        "text file(N)? ",yes) then                 \
        textfile=true : return
    if yesno%(crow+1,coloffset%,                   \
        "Replace record count ?"+                  \
        "                    ",yes)                \
        then fixhdr=true
    if fixhdr then                                 \
      if not yesno%(crow+1,coloffset%,             \
        "Use physical record count (Y),"+          \
        " or ask? (N) ",none)                      \
        then askhdr=true
    if yesno%(crow+1,coloffset%,                   \
        "Filter Control-Zs in the body "+          \
        "of the file? ",yes)                       \
        then zfilter=true
    if yesno%(crow+1,coloffset%,                   \
        "Filter other control characters"+         \
        " in file  ? ",yes)                        \
        then cfilter=true
    if (zfilter or cfilter) then                   \
        if yesno%(crow+1,coloffset%,               \
            "Ask before replacing control"+        \
            " characters  ? ",no)                  \
            then ask.replace=true
    return

read.header:
    rem dbase 521-byte header
    for d%=1 to dbheader%
        header.array%(d%)=get(infile%)
    next d%
    recs%=header.array%(3)*256+header.array%(2)
    recsize=header.array%(8)*256+header.array%(7)
    return

display.header:
    print
    if header.array%(1)<>dbasefile then            \
        call displ1(23,1,                          \
        "File is NOT marked as a "+                \
        "dBase II database") :                     \
        call syserr(24,1,                          \
        "Program behaviour may "+                  \
        "be 'unpredictable'!") :                   \
        call displ1(23,1," ") :                    \
        call displ1(24,1," ")
    print:print tab(coloffset%);                   \
                "Total records "; recs%
    print tab(coloffset%); "Last update    ";\
          str$(header.array%(4));"/";              \
          str$(header.array%(5));"/";              \
          str$(header.array%(6))
    print tab(coloffset%); "Record size    ";\
          recsize;" bytes"
    pos%=8
    print:print tab(coloffset%);                   \
                "Field Name   Type   Size"
    for n%=1 to 32
      fieldname$=""
      for o%=1 to 10
        if header.array%(pos%+o%)<>0 then          \
        fieldname$=fieldname$+                     \
        chr$(header.array%(pos%+o%))
      next o%
      spacing$=string$(15-len(fieldname$)," ")
      if (fieldname$<>"" and fieldname$<>cr)\
        then print tab(coloffset%);                \
        fieldname$;spacing$;                       \
        chr$(header.array%(pos%+12));              \
        "     ";header.array%(pos%+13)             \
      else                                         \
        n%=33
        pos%=pos%+16
    next n%
    print
check.ok:
    rem - routine to allow user to input new
    rem - header info to go here
    rem - (in case of complete corruption)
    rem - for the moment we will just check
```

dBase

FIXDBF.BAS *continued.*

```
    rem - whether recsize is ok
    if not yesno%(23,coloffset%,                \
      "Is the record size - "                   \
      +str$(recsize)+" bytes - okay?      "     \
      ,none) then                               \
      print : print tab(coloffset%);:input      \
"Actual record size?                            "\
      ;recsize
 return

ask.header:
    print tab(coloffset%);:                     \
    input                                       \
"New record count?                              ";xx%
    header.array%(2)=mod(xx%,256)
    header.array%(3)=int%(xx%/256)
    copyrecs%=xx%
    return

write.header:
    rem dbase 521-byte header
    for d%=1 to dbheader%
        put outfile%,header.array%(d%)
    next d%
    return

copy.database:
    dim a%(recsize)
    last.cpmeof=false
    print : print : print
    print tab(coloffset%-9);"working  ";
    while recount%<copyrecs%
      for x%=1 to recsize
        a%(x%)=get(infile%)
        if cfilter then
          if (a%(x%)<=control% or           \
            a%(x%)>maxasc%) and             \
            a%(x%)<>cpmeof                  \
            then gosub goask
        if zfilter then                     \
          if a%(x%)=cpmeof then             \
            gosub check.eof                 \
          else                              \
            last.cpmeof=false
      next x%
      recount%=recount%+1 : print "*"; :    \
      newline%=newline%+1 :                 \
      if newline%=working% then print:      \
         print tab(coloffset%);             \
         : newline%=0
      for x%=1 to recsize
        put outfile%,a%(x%)
      next x%
      rem - add cr/lf for text files
      if textfile then                      \
        put outfile%,13 :                   \
        put outfile%,10
    wend
    clean.finish=true
    return

check.eof:
    if last.cpmeof then                     \
        goto finito                         \
    else                                    \
        last.cpmeof=true: gosub goask
    return

goask:
    if a%(x%)=cpmeof then                   \
        zcount=zcount+1                     \
    else                                    \
        ccount=ccount+1
    if not ask.replace then                 \
        a%(x%)=32 : goto 100
    print:print : call displ1(24,1,         \
      "Found a control character in record "\
      +str$(recount%))
    for i%=1 to x%
      if a%(i%)>control% and a%(i%)<maxasc% \
        then print chr$(a%(i%));
      if a%(i%)<=control% then print " ^";  \
        chr$(a%(i%)+64);" ";
      if a%(i%)>maxasc% then print " ^";    \
        chr$(a%(i%)-64);" ";
    next i%
    print
    if yesno%(24,1,"Replace it ",yes) then  \
        gosub asknum
100 print : print tab(coloffset%);
    return

asknum:
    print
    input "Replace with what? (decimal) ";new%
    a%(x%)=new%
    return

fix.header:
    rem - we have to read through the file
    rem - once to determine size!
    rem - a calculation based on filesize
    rem - is inaccurate . . .
    print:print:print
    print tab(coloffset%);                  \
    "Analysing data file....please wait "
    if end # infile% then restart
    eof.markers%=0
    while true
      for r%=1 to recsize
        q%=get(infile%)
        if q%=cpmeof then                   \
          eof.markers%=eof.markers%+1
      next r%
      actual.recs=actual.recs+1
    wend
restart:
    actual.recs=actual.recs-                \
      (eof.markers%/recsize)
    copyrecs%=actual.recs
    rem - close file after dummy read
    rem - then re-open and re-position for
    rem - actual processing
    close infile%
    open in.file.name as infile%
    if end # infile% then finito
    for d%=1 to dbheader%
        q%=get(infile%)
    next d%
    header.array%(2)=mod(actual.recs,256)
    header.array%(3)=int%(actual.recs/256)
    return

print.stats:
    print:print:print
    call displ1(23,coloffset%,              \
        "Input File: "+in.file.name)
    print:print
    call displ1(23,coloffset%,str$(recs%)+  \
      " was the record count in file header")
    print:print
    if not askhdr then                      \
      call displ1(23,coloffset%,            \
      str$(recount%)+                       \
      " was the number of physical records")\
    else                                    \
```

```
        call displ1(23,coloffset%,          \
        str$(copyrecs%)+" records copied")
    if zfilter then
      if zcount>1 then print : print :      \
        call displ1(23,coloffset%,          \
        str$(zcount-1)+                     \
        " unwanted control-Zs were found")
    if cfilter then                         \
      if ccount>0 then print : print :      \
        call displ1(23,coloffset%,          \
        str$(ccount)+                       \
        " control characters (not "+        \
        "including ^Z) were found")
    print:print
    call displ1(23,coloffset%,              \
        "Output File: "+                    \
        out.file.name) : print
    if fixhdr then print:print:             \
      print tab(coloffset%);                \
      "**WARNING**": print tab(coloffset%); \
      "Record count has been rewritten,":   \
      print tab(coloffset%);                \
      "- there may be garbage records"      \
      : print tab(coloffset%);              \
      "at end of file." : print :           \
      print tab(coloffset%);                \
      "The new file should be PACKed":      \
      print
    return

finished:
    for i%=1 to x%
        put outfile%,a%(i%)
    next i%
    if textfile then put outfile%,13 :      \
        put outfile%,10
fini2:
    put outfile%,cpmeof
    close infile%,outfile%
    return

helpinfo:
    print:print "Help messages to be put here!!"
    stop

usage:
print
print tab(coloffset%); "Usage:"
print
print tab(coloffset%);"       FIXDBF, or"
print tab(coloffset%);                      \
      "       FIXDBF d:file1 d:file2"
print
print tab(coloffset%);"where:"
print tab(coloffset%);                      \
      "       file1 = source file, ";
print tab(coloffset%);"(* and ? not allowed)"
print tab(coloffset%);                      \
      "       file2 = destination file, ";
print tab(coloffset%);"(* and ? not allowed)"
print tab(coloffset%);                      \
      "       d:    = optional drive identifier"
print tab(coloffset%);                      \
      "i.e.   FIXDBF a:foobar.dbf b:fubar.dbf"
print
print tab(coloffset%);                      \
      "Source file MUST be a dBase II database"
stop

ofile.error:
    print
    print : call syserr(24,1,               \
    "Output file error - can't "+           \
    "open (or complete)"+out.file.name)
    stop

errhandler:
    call err.handler
    stop
end
```

FIND

This program was created using the dBase II MODIFY command. dBase II lacks a command to search for duplicate records, but this program will search a database file and locate them. In my example the search is on a field called 'code' (this field must already be sorted) but it could be on any field.

Peter Grosvenor
Yallambie, Vic

```
* FIND.CMD ..........................BY PETER.GROSVENOR. 20/1/85

* Find duplicate records based on the code

* save variables and restore them when done
SAVE TO temp

* start at the beginning of the file
GOTO TOP

* set up a loop to repeat until the whole file has been processed
* or the user decides to quit
STORE t TO more
DO WHILE more .AND. (.NOT. EOF)

* display something for the user to read
* while the program is searching for duplicates
ERASE
?
?
?
?
?
?
```

dBase

```
? "            I am Searching for duplicate records Master"

STORE !(code) TO oldcode

SKIP

IF oldcode = !(code) .AND. (.NOT. EOF)
? "please note duplicate code, and press any key to CONTINUE"
SET CONSOLE OFF
WAIT
SET CONSOLE ON
ENDIF

ENDDO
?
?
? "            I have finished Master"
?
?

RESTORE FROM temp

* End of story
```

DIAMOND MINER

This program lets you play Diamond Dan, a diamond miner. Your boss wants you to mine for as many diamonds as possible, and to speed you up a bit he has imposed a time limit within which to collect individual stones. The limit starts at 14 seconds, but if you get too fast — picking up a diamond in, say, five seconds — the boss may lower your time limit.

You must also be careful to collect only the white diamonds; the green ones are booby-trapped and will cost you a life. Watch out for rocks, as they will also reduce your life expectancy. The worst of all dangers are the walls: bumping into them will kill you instantly.

Added to all these dangers are red crosses which, when landed on, will move you randomly to another area of the mine.

With this game you can plug the joystick into either port. If you don't have a joystick, use the following keys instead:

A = left
D = right
W = up
X = down

The codes listed below substitute for graphics characters on the Commodore.

(C/RT) = Cursor Right
(C/DN) = Cursor Down
(RVON) = Reverse On
(RVOF) = Reverse Off
(PURP) = Purple
(LRED) = Light-red
(LGRN) = Light-green
(LBLU) = Light-blue
(YELO) = Yellow
(CLR) = Clear
(HOME) = Home

Paul Vandenberg
Cabramatta NSW

```
0 REM DIAMOND MINER BY PAUL VANDENBERG
1 POKE53280,0:POKE53281,0:PRINT"{PURP}"
2 CH=5:LY=20:LX=20
3 TI$="000000":TL=900
4 D$="{C/DN}":R$="{C/RT}"
5 JS=56320:GOSUB2000
6 S=54272:FORZ=STOS+24:POKEZ,0:NEXT:POKE
S+5,9:POKES+6,0:POKES+24,15
100 PRINT"{CLR}";
101 PRINT"{RVON}
                    ";
102 FORF=1TO20:PRINT" ";SPC(38);" ";:NEX
TF
103 PRINT"
          {RVOF}"
105 CL=1:FORF=1TO5:X=INT(RND(0)*38)+1:Y=
INT(RND(0)*20)+1:C=90:GOSUB10000
106 NEXTF
109 FORF=1TO15:A=PEEK(197):J1=PEEK(JS):J
2=PEEK(JS+1):POKES+4,32
110 POKES+1,50:POKES+4,129:POKES+4,0:IFT
I>=TLTHENGOSUB8000
111 CL=0:C=32:X=LX:Y=LY:GOSUB10000:IFA=9
ORJ1=126ORJ2=254THENLY=LY-1
112 IFA=230RJ1=125ORJ2=253THENLY=LY+1
113 IFA=100RJ1=123ORJ2=251THENLX=LX-1
114 IFA=180RJ1=119ORJ2=247THENLX=LX+1
115 P=LY*40+LX:L=PEEK(55296+P)
116 POKES+4,32:IFL=4THEN9000
117 IFL=6THENCH=CH-1:POKES+1,8:POKES+4,3
3:FORZ=1TO50:NEXTZ:IFCH=0THEN9000
118 IFL=1THENPT=PT+100:CT=TI:TI$="000000
":GOSUB5000:IFCT>360THENTL=TL-20
119 IFL=5THENCH=CH-1:POKES+1,8:POKES+4,3
3:FORZ=1TO50:NEXTZ:IFCH=0THEN9000
120 IFL=10THENGOSUB5050:LX=INT(RND(0)*38
)+1:LY=INT(RND(0)*20)+1
121 CL=14:C=42:X=LX:Y=LY:GOSUB10000
122 IFTI$="000015"THENGOSUB8000
124 PRINT"{HOME}{RVON}{PURP}"TI$;TAB(10)
;"SCORE";PT;TAB(26);"MEN LEFT";CH
129 NEXTF
130 CL=1:FORF=1TO1:X=INT(RND(0)*38)+1:Y=
INT(RND(0)*20)+1:C=90:GOSUB10000
131 CL=6:FORF=1TO4:X=INT(RND(0)*38)+1:Y=
INT(RND(0)*20)+1:C=224:GOSUB10000:NEXTF
132 CL=5:X=INT(RND(0)*38)+1:Y=INT(RND(0)
*20)+1:C=90:GOSUB10000
133 IFRND(0)>.8THENCL=10:X=INT(RND(0)*38
)+1:Y=INT(RND(0)*20)+1:C=86:GOSUB10000
160 GOTO109
2000 REM TITLE PAGE
2001 PRINT"{CLR}";:FORZ=1TO5:PRINTD$;:NE
XTZ
2002 PRINT"{PURP}";
2003 PRINT"{RVON}  ◥{C/RT}   {C/RT}◤ ◥{C
/RT}◥{C/RT}{C/RT}◤{C/RT}◤ ◥{C/RT}◥{C/RT}
{C/RT} {C/RT}  ◥"
2004 PRINT"{RVON} {C/RT} {C/RT}{C/RT} {C
/RT}{C/RT} {C/RT} {C/RT} {RVOF}╲╱{RVON}
{C/RT} {C/RT} {C/RT} {RVOF}╲{RVON}{C/RT}
 {C/RT} {C/RT} "
2005 PRINT"{RVON} {C/RT} {C/RT}{C/RT} {C
/RT}{C/RT}   {C/RT} {C/RT}{C/RT} {C/RT}
{C/RT} {C/RT} {C/RT}{RVOF}╲{RVON} {C/RT}
 {C/RT} "
2006 PRINT"{RVON} {C/RT} {C/RT}{C/RT} {C
/RT}{C/RT} {C/RT} {C/RT} {C/RT}{C/RT} {C
/RT} {C/RT} {C/RT} {C/RT}{C/RT} {C/RT} {
C/RT} "
2007 PRINT"{RVON}  {RVOF}◤{RVON}   {C/R
T} {C/RT} {C/RT} {C/RT}{C/RT} {C/RT}{RVO
F}◥{RVON} {RVOF}◤{C/RT}{RVON} {RVOF} {C/
RT}{RVOF}◥ {RVON}  {RVOF}◤"
2008 PRINT:PRINT
2009 PRINTTAB(10);"{RVON}◥{C/RT}{C/RT}◤(
C/RT}   {C/RT}◥{C/RT}{C/RT} {C/RT}   {C/
RT}  ◥"
2010 PRINTTAB(10);"{RVON} {RVOF}╲╱{RVON}
 {C/RT}{C/RT} {C/RT}{C/RT} {RVOF}╲{RVON}
{C/RT} {C/RT} {C/RT}{C/RT}{C/RT} {C/RT}
"
2011 PRINTTAB(10);"{RVON} {C/RT}{C/RT} {
C/RT}{C/RT} {C/RT}{C/RT} {C/RT}{RVOF}╲{R
VON} {C/RT}  {C/RT}{C/RT}  {RVOF}◤"
2012 PRINTTAB(10);"{RVON} {C/RT}{C/RT} {
C/RT}{C/RT} {C/RT}{C/RT} {C/RT}{C/RT} {C
/RT} {C/RT}{C/RT}{C/RT} {RVOF}╲{RVON}"
2013 PRINTTAB(10);"{RVON} {C/RT}{C/RT} {
C/RT}   {C/RT} {C/RT}{C/RT}{RVOF}◥ {RVON
}   {C/RT} {C/RT}{RVOF}╲"
2015 PRINT:PRINT"{C/RT}{C/RT}{C/RT}{C/RT
}{C/RT}{LGRN}BY PAUL VANDENBERG"
2016 PRINT:PRINT"{LRED}PRESS A KEY TO ST
ART{PURP}"
2017 GETA$:IFA$=""THEN2017
2018 RETURN
5000 POKES+1,50:POKES+4,17:POKES+4,16:FO
RZ=1TO20:NEXTZ


5001 POKES+1,60:POKES+4,17:POKES+4,16:FO
RZ=1TO20:NEXTZ
5002 RETURN
5050 FORF=40TO70STEP2:POKES+1,F:POKES+4,
33:NEXTF:POKES+4,32:RETURN
7999 REM NEXT LEVEL MARKER""
8000 REM TIME UP
8001 PRINT"{HOME}";:FORZ=1TO23:PRINTD$;:
NEXTZ:PRINT"{PURP}YOU WERE A BIT SLOW"
8002 POKES+1,6:POKES+4,33:FORZ=1TO2000:N
EXTZ:CH=CH-1:IFCH=0THEN9000
8003 PRINT"{C/UP}                    {HOM
E}":POKES+4,32
8004 TI$="000000":RETURN
8500 RETURN
9000 PRINT"{CLR}YOU'RE DEAD!!"
9001 GETA$:IFA$<>""THEN9001
9002 PRINT"{LBLU}YOU'VE ACCUMULATED A TO
TAL SCORE OF"
9003 PRINT:FORZ=1TO20:PRINT"{C/RT}{C/RT}
{C/RT}{C/RT}{C/UP}{YELO}";PT:FORG=1TO40:
NEXTG:PRINT"{C/RT}{C/RT}{C/RT}{C/RT}{C/U
P}{GRN}";PT
9004 POKES+1,5:POKES+4,33:POKES+4,32
9005 FORG=1TO38:NEXTG:NEXTZ
9006 PRINT:PRINT"{LGRN}WOULD YOU LIKE AN
OTHER GO? (Y/N)"
9007 GETA$:IFA$=""THEN9007
9008 IFA$="Y"THENCLR:GOTO1
9009 END
10000 P=Y*40+X:POKEP+1024,C:POKE55296+P,
CL:RETURN
```

Commodore 16

COMPATIBILITY ANALYSIS

The program displays a compatibility table for two people based on the biorhythmic cycle theory. The table consists of physical cycle, sensitivity cycle and cognitive cycle compatibility.

Although it was written on a C16, the program should also run without modification on the Plus Four.

Mark Wilkinson
Heyfield VIC

```
10 REM ***************************
20 REM *      COMMODORE 16       *
30 REM * COMPATABILITY ANALYSIS *
40 REM *   AUTHOR:M.WILKINSON    *
50 REM ***************************
60 REM
70 DIMA1(30),B1(30),M$(12),D$(7),A(12)
80 COLOR0,3,2:COLOR4,6,3
90 FORI=1TO12:READA(I):NEXT
100 DATA0,31,59,90,120,151,181,212,243,273,304,334
110 GOSUB800
120 Y=0:FO$="%###.##"
130 Y=Y+1
140 INPUT"■■■NAME OF PERSON 1.....■";W$
150 INPUT"■BIRTHDAY (DD,MM,YYYY).■";D,M,Y
160 E1=M:F1=D:G1=Y:GOSUB600
170 Z2=T:K1=J+1
180 INPUT"■■■NAME OF PERSON 2.....■";X$
190 INPUT"■BIRTHDAY (DD,MM,YYYY).■";D,M,Y
200 E2=M:D2=D:G2=Y:GOSUB600
210 P2=ABS(Z2-T):K2=J+1
220 PRINT"■";
230 PRINTTAB(8)"■┌──────────────────────────┐"
240 PRINTTAB(8)"│■■COMPATABILITY ANALYSIS■■│"
250 PRINTTAB(8)"└──────────────────────────┘■"
260 PRINT TAB((40-LEN(W$))/2);W$
270 PRINT TAB((40-LEN(X$))/2);X$;"■"
280 PRINT
290 PRINT"■"W$":■";
300 J=K1:GOSUB790:PRINT",";
310 M=E1:GOSUB780
320 PRINTF1","G1
330 PRINT"■■"X$":■";
340 J=K2:GOSUB790:PRINT",";
350 M=E2:GOSUB780
360 PRINTD2","G2"■■"
370 Z=P2
380 P3=ABS(INT(((Z/23)-INT(Z/23))*23))
390 S3=ABS(INT(((Z/28)-INT(Z/28))*28))
400 C3=ABS(INT(((Z/33)-INT(Z/33))*33))
410 P5=ABS(100-((2*P3)*(100/23)))
420 S5=ABS(100-((2*S3)*(100/28)))
430 C5=ABS(100-((2*C3)*(100/33)))
440 PRINTTAB(9)"■┌──────────────────────┐"
450 PRINTTAB(10)"■PHYSICAL.....■";
460 PRINT USING FO$;P5
470 PRINTTAB(10)"■SENSITIVITY..■";
480 PRINT USING FO$;S5
490 PRINTTAB(10)"■COGNITIVE....■";
500 PRINT USING FO$;C5
510 PRINTTAB(9)"■└──────────────────────┘"
520 A5=(P5+S5+C5)/3
530 PRINTTAB(10)"■AVERAGE......■";
540 PRINT USING FO$;A5
550 PRINTTAB(9)"■┌──────────────────────┐"
560 PRINT"■■■■■■"
570 PRINT"■■   PRESS ANY KEY FOR ANOTHER ANALYSIS  ■";
580 GETKEY A$
590 PRINT"■";:RUN
600 Y1=Y-1800
610 Q1=INT(Y1/4)
```

```
620 Q2=INT(Q1/25)
630 Q3=INT((Y1+200)/400)
640 K=0
650 IFQ1*4<>Y1THEN690
660 IFQ2*100<>Y1THEN690
670 IFQ3*400-200<>Y1THEN690
680 K=1
690 T=365*Y1+Q1-Q2+Q3-K
700 T=T+A(M)+D-1
710 IFM<3THEN730
720 T=T+K
730 IFINT(Y1/4)<>Y1/4THEN760
740 IFM>2THEN760
750 T=T-1
760 J=T-7*INT(T/7)
770 RETURN
780 PRINTM$(M);:RETURN
790 PRINTD$(J);:RETURN
800 FORJ=1TO12:READM$(J):NEXT
810 FORJ=1TO7:READD$(J):NEXT:RETURN
820 DATAJAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP
830 DATAOCT,NOV,DEC
840 DATAWEDNESDAY,THURSDAY,FRIDAY,SATURDAY
850 DATASUNDAY,MONDAY,TUESDAY
860 REM
870 REM
880 REM-----------------
890 REM  CONTROL CODES
900 REM IN THIS LISTING
910 REM-----------------
920 REM "□ = CLEAR
930 REM "■ = DOWN
940 REM "■ = BLU
950 REM "■ = PUR
960 REM "■ = ORNG
970 REM "■ = GRN
980 REM "■ = FLASH ON
990 REM "■ = FLASH OFF
1000 REM-----------------
```

Microbee

MOTORBIKE TRIALS

Motorbike Trials simulates a rider travelling a twisting 100 kilometre course.

The rider starts at the top of the screen facing down. The 'A' key moves him to the player's right and the '\' key moves him to the player's left.

Every few kilometres the rider moves toward the bottom of the screen and is followed by another rider (you always control the front one). If your rider reaches the bottom of the screen, you've completed the course — it's very hard!

Good luck ... and watch for slow-moving cars.

Ken Rowe
Dernancourt SA

```
00090 GOTO 700
00095 CLEAR:RESTORE
00100 P=63488+49*16
00110 FOR A=P TO P+16*8-1
00120 READ B:POKE A,B
00130 NEXT A
00140 DATA 36,255,255,255,60,36,189,189,126,126,60,60,189,255,189,24
00150 DATA 15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15
00160 DATA 240,240,240,240,240,240,240,240,240,240,240,240,240,240,240,240
00170 DATA 0,16,16,16,40,56,124,108,186,68,56,16,16,16,0,0
00180 DATA 0,0,1,2,4,30,46,73,50,28,32,32,64,128,0,0
00190 DATA 0,0,128,64,32,32,124,114,73,42,20,4,2,1,0,0
00200 DATA 8,137,73,34,20,93,46,86,28,99,22,236,26,41,68,32
00210 DATA 15,15,15,15,240,240,240,240,15,15,15,15,240,240,240,240
00220 CLS: R=1:N=33:E=5
00230 GOSUB 520
00240 IF RND*1<.2 THEN LET R=INT(RND*3)
00250 K=K-1+R:Z=0
00280 IF PEEK(258)=28 THEN LET N=N+1:Z=2 ELSE IF PEEK(258)=1 THEN LET N=N-1:Z=1
00290 PCG:POKE 61440+(Q*64)+N,180+Z:NORMAL
00300 IF K>20 THEN LET K=20 ELSE IF K<-29 THEN LET K=-29
00320 IF RND*2<.112 THEN 460
00330 PRINT TAB(30+K);:PCG:PRINT"22";:NORMAL:PRINT SPC(E);:PCG:PRINT"33":NORMAL
00340 IF PEEK(61440+(64*Q)+N)<180 AND PEEK(61440+(64*Q)+N)>175 THEN 370
00350 S1=S1+1:IF S1=100 THEN GOSUB 800
00360 GOTO 240
00370 POKE 61440+(64*Q)+N,183:FOR I=1 TO 22:IF RND*1<.55 THEN OUT 2,59:OUT 2,65:
NEXT I ELSE NEXT I
00380 PLAY 0,1;22,5;19,5;22,5;19,5;22,5;19,5
00390 F=Q*100+INT(S1):CURS1:PRINT"YOU COMPLETED ";INT(FLT(F)/1500*100);" Km OF A
 100 Km COURSE."\"AND PICKED UP ANOTHER ";Q;" RIDERS"\"1] ANOTHER GAME"\"2] END"
00400 A1$=KEY$:IF A1$="" THEN 400
00410 IF A1$="1" THEN 95 ELSE IF A1$="2" THEN PRINT\"BYE":END
00420 GOTO 400
00430 PRINT TAB(30+K);:PCG:PRINT"22";:FORI=1TOE:PRINT"8";:NEXTI:PRINT"33":NORMAL
00440 PLAY5,5;5,5;7,5;9,5;7,5;5,5;5,5:CURS1:PRINT"YOU HAVE COMPLETED THE 100 Km
COURSE WITH 14 RIDERS"\"THAT IS AN EXCELLENT ACHIEVEMENT!"\"1] ANOTHER GAME"\"2]
 END":GOTO400
00450 END
00460 IF RND*1<.35 THEN 490
00470 W=INT(RND*(FLT(E))+1):PRINTTAB(30+K);:PCG:PRINT"22";:NORMAL:PRINT SPC(W-1)
;:PCG:PRINT"1";:NORMAL:PRINTSPC(E-W);:PCG:PRINT"33":NORMAL
00480 GOTO340
00490 IF RND*1<.5 THEN LET E=E+1 ELSE LET E=E-1
00500 IF E<4 THEN LET E=4 ELSE IF E>8 THEN LET E=8
00510 GOTO 330
00520 NORMAL: FOR I=1 TO 15:IF I=2THEN570 ELSE PRINT TAB(30);:PCG:PRINT"22";:NOR
MAL:PRINT SPC(5);:PCG:PRINT"33":NORMAL:NEXT I
00530 POKE 61440+N,180
00535 CURS5,2:PRINT"<== 'A'";:CURS54,2:PRINT"'\' ==>";:CURS15,16
00540 FOR I=1 TO5:FOR D=1 TO 25:OUT 2,59:OUT 2,65:FOR G=1 TO D
00550 NEXT G:NEXTD:NEXTI
00560 RETURN
00570 PRINT TAB(30);:PCG:PRINT"228888833":NORMAL:NEXT I
00700 CLS:PLAY 16,2;16,2;13,4;16,4;18,2;18,2;16,4;18,4;18,4;20,2;18,2;16,2;16,4
00710 INPUT "   INSTRUCTIONS ? ";T1$
00720 IF T1$="YES"ORT1$="yes"ORT1$="y"ORT1$="Y"THEN730ELSE100
00730 CLS: PRINT"                              ** MOTORBIKE TRIALS **"\"
      ****************"\
00740 PRINT"  You are set a difficult 100 km course."\"Along the way you are joi
ned by more riders."\"You will lead these riders through the winding corners"
00750 PRINT"And past slower cars."\\"The direction of the motorbike is to the bo
ttom of the screen"\"'A' moves the bike to your left & '\' to your right."
00760 PRINT\"The road changes width as well as direction."\\"If you can pick up
another 14 riders and reach the bottom of "\"the screen, you have completed the
course (very hard!!)"
00770 INPUT"                                          HIT <<RETURN>> TO START";T4$
00780 GOTO 100
00800 Q=Q+1:IF PEEK(258)=28THENLETN=N+1 ELSE IF PEEK(258)=1THENLETN=N-1
00810 S1=0 :IF Q=15 THEN 430
00820 RETURN
```

CLOCK

Here is a program for all the egg burners among you. Just set the clock and alarm for perfect eggs, or use the stop watch for timing those three-hour chess games. The letter 'S' will stop and start the stop watch; press 'R' after the watch is stopped to return to the menu.

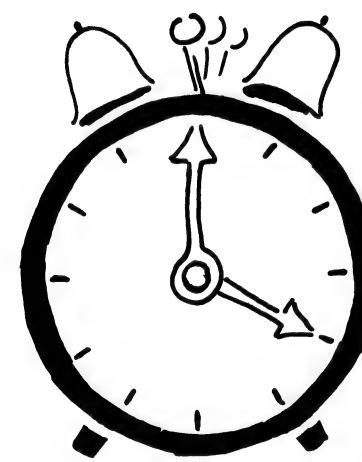
The program is quite easy to convert to other machines, given that 'Lores' and 'Hires' select graphics modes for the borders around the outside of the menu and the introduction. All 'Plots' can be left out, or your own computer's graphics commands may be substituted. CLS clears the screen. The Key$ is the same as Inkey$ on most other machines. The rest is fairly universal.

Jeremy Fenton
Lismore NSW

```
00100 REM ################################################
00110 REM ##  CLOCK  BY J. FENTON                       ##
00120 REM ##  FOR PUBLIC USE                            ##
00130 REM ##  LAST UPDATE  2/3/85                       ##
00140 REM ################################################
00150 LET X=0
00160 REM ################ INTRODUCTION ###################
00170 CLS
00180 HIRES
00190 PLOT 0,0 TO 0,255 TO 511,255 TO 511,0 TO 0,0
00200 PLOT 100,150 TO 50,150 TO 50,200 TO 100,200
00210 PLOT 120,200 TO 120,150 TO 170,150
00220 PLOT 190,200 TO 190,150 TO 240,150 TO 240,200 TO 190,200
00230 PLOT 310,200 TO 260,200 TO 260,150 TO 310,150
00240 PLOT 330,200 TO 330,150:PLOT 330,175 TO 390,200: PLOT 331,176 TO 390,150
00250 CURS 40,15:PRINT"PRESS ''RETURN''"
00260 A3$=KEY$:IF A3$="" THEN GOTO 260
00270 GOTO 720
00280 REM ################ SET CLOCK ######################
00290 CLS
00300 CLS:CURS 21,7:INPUT"ENTER TIME)-HOURS"H
00310 CURS 32,8:INPUT"-MINUTES"M
00320 CURS 32,9:INPUT"-SECONDS"S
00330 RETURN
00340 REM ################ CLOCK ####################
00350 CLS
00360 GOSUB 570
00370 GOSUB 620
00380 GOSUB 630
00390 GOSUB 640
00400 S=S+1
00410 FOR A0=1 TO 31 STEP.1087899
00420 NEXT A0
00430 GOSUB 640
00440 IF S<60 THEN 400
00450 S=0
00460 GOSUB 640
00470 M=M+1
00480 GOSUB 630
00490 IF M<60 THEN 400
00500 M=0
00510 GOSUB 630
00520 H=H+1
00530 GOSUB 620
00540 IF H<13 THEN 400
00550 H=1
00560 GOTO 370
00570 CURS 20,7:PRINT"HOURS  MINUTES  SECONDS"
```

Microbee

```
00580 CURS 30,15:PRINT"PRESS ''R'' FOR MENU"
00590 IF Q=H AND K=M AND V=S THEN PLAY 1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1
00600 RETURN
00610 REM ++++++++++++++++ ALARM ++++++++++++++++++++
00620 CURS 21,8:PRINT H" ":RETURN
00630 CURS 29,8:PRINT M" ":RETURN
00640 CURS 37,8:PRINT S" ":IF Q=H AND K=M AND V=S THEN PLAY 1;1;1;1;1;1;1;1;1;1;1;1
00650 A5$=KEY$:IF A5$="R" OR A5$="r" THEN GOTO 720
00660 RETURN
00670 REM ++++++++++++++++ SET ALARM ++++++++++++++++++++
00680CLS: CURS21,8:INPUT"HOUR >";Q
00690 CURS 21,9:INPUT"MINUTES >";K
00700 CURS 21,10: INPUT"SECONDS >";V
00710 RETURN
00720 CLS
00730 REM ++++++++++ MENU ++++++++++++++++++++++++++++
00740 LORES:PLOT 0,0 TO 0,47 TO 127,47 TO 127,0 TO 0,0
00750 CURS 30,5:PRINT"MENU"
00760 CURS 10,7: PRINT"SET THE CLOCK : S"
00770 CURS 10,9: PRINT"STOP WATCH    : W"
00780 CURS 10,11:PRINT"SET THE ALARM : A"
00790 CURS 10,13:PRINT"START CLOCK   : C"
00800 A5$=KEY$:IF A5$="" THEN GOTO 800
00810 IF A5$="S" OR A5$="s" THEN GOSUB 300
00820 IF A5$="a" OR A5$="A" THEN GOSUB 680
00830 IF A5$="C" OR A5$="c" THEN GOTO 350
00840 IF A5$="W" OR A5$="w" THEN GOTO 870
00850 GOTO 720
00860 REM ++++++++++++++ STOP WATCH ++++++++++++++++++++
00870 CLS
00880 CURS 20,6:PRINT" MINUTES   SECONDS ":GOTO 900
00890 FOR P=1 TO 800:NEXT P
00900 A5$=KEY$:IF A5$="S" OR A5$="s" THEN GOTO 970
00910 LET Z=Z+1
00920 CURS 34,8:PRINT Z
00930 CURS 25,8: PRINT X
00940 A5$=KEY$:IF A5$="S" OR A5$="s" THEN GOTO 970
00950 IF Z=60 THEN LET Z=0: X=X+1: GOTO 870
00960 GOTO 890
00970 A5$=KEY$:IF A5$="" THEN GOTO 970
00980 IF A5$="S" OR A5$="s" THEN GOTO 950
00990 IF A5$="R" OR A5$="r" THEN LET Z=0:LET X=0:GOTO 720
01000 GOTO 970
```



VZ200

NUMBER SEQUENCE

This program prints various sequences of numbers, each ending with a blank. You must enter the next number in the sequence — the computer indicates if your entry was correct.

A series of ten questions is asked, then your score is given.

Because the program is written in standard Microsoft BASIC, it should be easily transported to other computers. The random number statements in lines 120-140 may need modification, according to your particular version of BASIC.

Ian Thompson
Collaroy Plateau NSW

```
1 '*****************************
2 '*      NUMBER SEQUENCE      *
3 '* FOR THE UNEXPANDED VZ-200 *
4 '* IAN THOMPSON - COLLAROY   *
5 '*****************************
10 CLS:PRINT@104,"NUMBER SEQUENCE"
12 PRINT@325,"IAN THOMPSON,COLLAROY"
15 PRINT@485,"PRESS ANY KEY TO START"
20 IF INKEY$="" THEN 20
21 IF INKEY$="" THEN 20
25 CLS:PRINT"         NUMBER SEQUENCE":PRINT
30 PRINT"THIS PROGRAM WILL PRINT VARIOUS"
35 PRINT"SEQUENCES OF NUMBERS, EACH      "
40 PRINT"ENDING WITH A BLANK (----)."
45 PRINT"WHEN YOU SEE A '?', TYPE IN THE"
50 PRINT"NUMBER THAT YOU THINK THE "
55 PRINT"COMPUTER MIGHT HAVE PRINTED IN "
60 PRINT"PLACE OF THE BLANK."
70 PRINT
75 PRINT"*********************************"
80 LET R=0
90 LET W=0
100 FOR I=1 TO 10
110 PRINT"PROBLEM";I
120 LET A=INT(10*RND(0)+1)
130 LET B=INT(10*RND(0)+1)
140 LET G=RND(3)
150 IF A>B THEN 285
160 IF G=1 THEN 170
162 IF G=2 THEN 210
164 IF G=3 THEN 250
170 LET X=2*A+3*B
180 PRINT A;",";B;",";A+B;",";A+2*B;", ----";
190 INPUT Y
200 GOTO 410
210 LET X=A*A*B*B*B
220 PRINT A;",";B;",";A*B;",";B*A*B;", ----";
230 INPUT Y
240 GOTO 410
250 LET X=-B
260 PRINT A;",";B;",";B-A;",";-A;", ----";
270 INPUT Y
280 GOTO 410
285 IF G=1 THEN 300
290 IF G=2 THEN 340
300 LET X=A*5
310 PRINT A;",";2*A;",";3*A;",";4*A;", ----";
320 INPUT Y
330 GOTO 410
340 LET X=16*A
350 PRINT A;",";2*A;",";4*A;",";8*A;", ----";
360 INPUT Y
410 IF X=Y THEN 450
420 PRINT"NO; THE COMPUTER'S SEQUENCE HAS ";X;"."
430 LET W=W+1
440 GOTO 470
450 PRINT"THAT'S RIGHT!"
460 LET R=R+1
470 PRINT
480 NEXT I
485 SOUND 15,5
490 PRINT"================================="
500 PRINT"SCORE: ";R;" RIGHT,";W;" WRONG
505 PRINT:PRINT"================================="
510 PRINT" PRESS <SPACE> FOR ANOTHER SET   OF";
520 PRINT" QUESTIONS."
530 A$=INKEY$:IF A$ <> " "THEN 530
535 RUN
```

TRS-80

MAILIST

Mailist is a mailing list program for TRS-80 Models I, III and IV with 16 Kbytes of RAM. It stores up to 250 names and addresses on cassette tape for later retrieval and printing.

For those of you who have machines with 48 KBytes of RAM, a version for up to 1100 names and addresses with a built-in lowercase driver is available for $10, cassette supplied.

Both programs are menu-driven and a help sequence is included in the program (function 8). If you have any questions or wish to purchase the 48 Kbyte version, write to David Minehan, 64 Young Road, Lambton East 2299.

David Minehan
Lambton East, NSW

```
10 ' MAILIST.
20 ' PROGRAMMER: DAVID KEITH MINEHAN.
30 ' SYSTEM: TRS-80/SYSTEM-80.
40 ' RAM: 16K.
50 ' LANGUAGE: BASIC.
60 ' PROGRAM DESIGNATION: BUSINESS.
70 '
80 ' PROGRaM DESIGNED TO TAKE NAMES,
90 ' ADDRESSES, AND PHONE NUMBERS &
100 ' STORE THEM ON CASSETTE-TAPE
110 ' DATA FILE(S).
120 '
220 CLEAR5500:CLS:POKE16396,23
    :DIMN$(250),R$(250),A$(250),P$(250)
230
    PRINT TAB(30);"MAILIST":PRINT:PRINT: ' ***MAIN MENU*** '
240
    PRINTTAB(10)"CREATE FILE(S).....1"
250 PRINTTAB(10)"VIEW   FILE(S).....2"
260 PRINTTAB(10)"UPDATE FILE(S).....3"
270 PRINTTAB(10)"SAVE   FILE(S).....4"
280 PRINTTAB(10)"CALL   FILE(S).....5"
290 PRINTTAB(10)"VERIFY FILE(S).....6"
300 PRINTTAB(10)"PRINT  FILE(S).....7"
310 PRINTTAB(10)"HELP.....8"
320 PRINTTAB(10)"QUIT.....9"
330 PRINT:PRINTTAB(10)"WHICH FUNCTION DO YOU WISH TO EXECUTE";:INPUTQ
340 ONQGOTO410,510,610,710,810,910,1010,1110,9999
350 CLS:GOTO230:    ' ***MENU END*** '
410 CLS:INPUT"WHEN READY, HIT <ENTER> (TO CLOSE FILE TYPE 9999 FOR NAME)";X:CLS
420 FORI=1TO250:PRINT:PRINT"ENTER NAME (LAST FIRST, NO COMMAS PLEASE)"
430 PRINT"THEN HIT THE <ENTER> KEY";:INPUTN$(I)
440 IFN$(I)="9999"THENP1=I:GOTO490 '
445 PRINT"ENTER NAME OF REPRESENTITIVE (<ENTER> FOR NIL)"
446 INPUTR$(I)
450 INPUT"ENTER STREET NUMBER & NAME (NO COMMAS)";A$(I)
460 INPUT"ENTER SUBURB/CITY & POSTCODE";P$(I)
470 IFFRE(X)<250GOTO490:IFFRE(X)=250THENPRINT:PRINT"FILE FULL"
    :PRINT
    :INPUT"PRESS <ENTER> TO RETURN TO MENU";X
    :CLS
    :GOTO230
480 NEXT
490 PRINT"FILE CLOSED -- ":INPUT"TO SEE MENU, HIT <ENTER>";X:CLS:GOTO230
510 CLS:FORI=1TOP1:PRINTN$(I):PRINTR$(I):PRINTA$(I):PRINTP$(I):PRINT:NEXT
520 PRINT:INPUT"TO RETURN TO MENU, HIT <ENTER>";X:CLS:GOTO230
610 CLS:PRINT"ENTER THE NAME FOR THE LINE YOU WISH TO CHANGE (NO COMMAS)"
620 INPUTN$
630 FORI=1TOP1:IFN$=N$(I)GOTO670
640 NEXT
650 PRINT:PRINT"NAME NOT IN FILE -- ":GOTO690
660 PRINT
    :PRINT"ENTER THE CORRECTED INFO. : NAME, ADDRESS, PHONE --"
670 INPUTN$(I),R$(I),A$(I),P$(I)
680 PRINT:PRINT"THE FILE NOW READS: "
    :PRINTN$(I):PRINTR$(I):PRINTA$(I):PRINTP$(I):PRINT
690 INPUT"FOR ANOTHER CORRECTION, TYPE 1, OTHERWISE TYPE 0";X
    :IFX=1THEN610ELSECLS:GOTO230
710 CLS:
    INPUT"PREPARE CASSETTE + PLAYER, WHEN READY HIT <ENTER>";X
```

```
720 PRINT:PRINT"COPYING..."
730 PRINT #-1,P1
740 FORI=1TOP1:PRINT #-1,N$(I),R$(I),A$(I),P$(I):NEXT
750 PRINT:PRINT"COMPLETE -- NOTE TAPE LOCATION PLEASE -- "
    :PRINT
760 INPUT"TO RETURN TO MAIN MENU, HIT <ENTER>";X
    :CLS:GOTO230
810 CLS:
    INPUT"PREPARE CASSETTE + RECORDER, WHEN READY HIT <ENTER>";X    :PRINT
820 PRINT"INPUTING..."
830 INPUT#-1,P1
840 FORI=1TOP1:INPUT#-1,N$(I),R$(I),A$(I),P$(I):NEXT:PRINT
850 INPUT"INPUTING OF DATA COMPLETE. TO SEE MENU PRESS <ENTER>";    X:CLS:GOTO23
0
910 CLS:PRINT"FILE VERIFICATION:":PRINT:PRINT
920 INPUT"NAME FOR SEARCH";N$:PRINT
930 FORI=1TOP1:IFN$=N$(I)THENPRINT"FILE FOUND -- ":GOTO960
940 NEXT
950 PRINT"FILE NOT FOUND -- ":INPUT"PRESS <ENTER> TO CONTINUE";
    X:CLS:GOTO230
960 PRINT:PRINTN$(I):PRINTR$(I):PRINTA$(I):PRINTP$(I)
        :INPUT"DO YOU WANT THE NAME PRINTED (Y/N)";X$
        :PRINT
        :IFX$="Y"THENLPRINTN$(I):LPRINTCHR$(24):LPRINTR$(I)
        :LPRINTCHR$(24):LPRINTA$(I):LPRINTCHR$(24)
        :LPRINTP$(I)
970 INPUT"PRESS 1 TO CONTINUE, 0 TO RETURN TO MENU";X
    :IFX=1THEN900ELSECLS:GOTO230

1005 CLS
1010 CLS:PRINT"PRINTING...":PRINT
1020 FORI=1TOP1:LPRINTN$(I):LPRINTR$(I):LPRINTA$(I):LPRINTP$(I):LPRINTCHR$(194):
NEXT
1030 PRINT:PRINT"PRINTING COMPLETE.":PRINT
1050 INPUT"TO SEE MENU HIT <ENTER>";X
    :CLS:GOTO230
1110 CLS:PRINTTAB(30);"HELP":PRINT:PRINT
1120 PRINT"CREATE ADDRESS FILE(S):

THIS PROCEDURE ALLOWS THE USER TO CREATE THE FILES TO BE USED
IN STORING THE NAMES, ADDRESSES, AND PHONE NUMBERS."
1130 PRINT
1140 PRINT"VIEW ADDRESS FILE(S):

THIS PROCEDURE ALLOWS THE USER TO SEE THE FILES EITHER JUST
CREATED, OR JUST TRANSFERED INTO MEMORY FROM TAPE."
1150 PRINT:INPUT"TO CONTINUE, PRESS 1, ELSE PRESS 0 TO RETURN
TO MAIN MENU";X
        :IFX=0THENCLS:GOTO230
        :IFX=1THENGOTO1160
1160 CLS:PRINT"UPDATE ADDRESS FILE(S):

THIS PROCEDURE ALLOWS THE USER TO CHANGE THE FILES JUST CREA-
TED (FOR TYPO'S), OR FROM TAPE, TO MAKE CHANGES TO PHONE NUM-
BERS OR ADDRESSE(S) AS NEEDED."
1170 PRINT
1180 PRINT"SAVE ADDRESS FILE(S):

THIS PROCEDURE ALLOWS THE USER TO SAVE THE FILE(S) JUST CREATED
OR THE CHANGES JUST MADE."
```

TRS-80

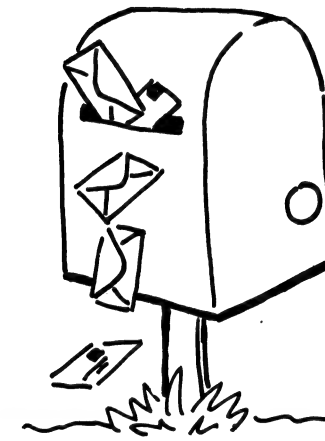
Mailist continued.

```
1190 PRINT
1200 PRINT:INPUT"TO CONTINUE, PRESS 1, ELSE PRESS 0 TO RETURN
TO MAIN MENU";X
          :IFX=0THENCLS:GOTO230
          :IFX=1THEN1210
1210 CLS:PRINT"CALL ADDRESS FILE(S):
THIS PROCEDURE ALLOWS THE USER TO CALL THE ADDRESS FILE(S) FROM
CASSETTE-TAPE DATA FILES."
1220 PRINT
1230 PRINT"VERIFY ADDRESS FILE(S):
THIS PROCEDURE ALLOWS THE USER TO VERIFY THE ADDRESS FILE(S).
THIS IS USED WHEN THE USER WISHES TO SEARCH FOR A PARTICULAR
NAME, ADDRESS AND PHONE NUMBER."
1240 PRINT:INPUT"TO CONTINUE, PRESS 1, ELSE PRESS 0 TO RETURN
TO MAIN MENU";X
          :IFX=0THENCLS:GOTO230
          :IFX=1THENGOTO1250
1250 CLS:PRINT"PRINT ADDRESS FILE(S):

THIS FUNCTION ALLOWS THE USER TO GET A HARD COPY OF THE FILES
EITHER ON PRINTER PAPER, OR STICKY LABELS FOR ENVELOPES."
1260 PRINT
1270 PRINT"HELP:

THIS FUNCTION."
1280 PRINT
1290 PRINT"QUIT:

END THE PROGRAM."
1300 PRINT
1310 INPUT"TO RETURN TO MENU, PRESS <ENTER>";X
     :CLS
     :GOTO230
9999 CLS:
      :POKE16396,201
      :END
```



TRS-80

MATH LAUNCH

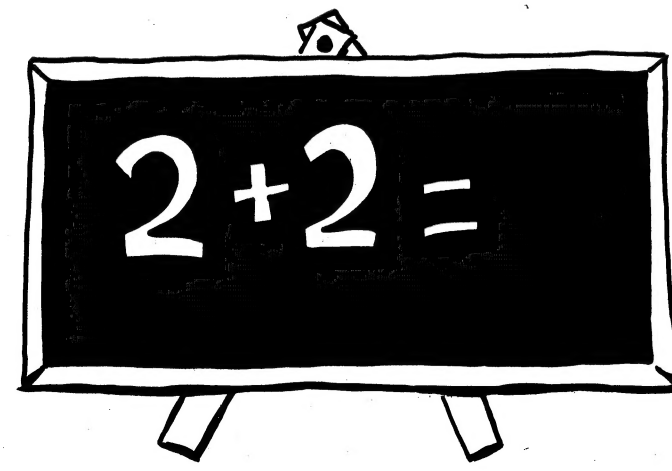
Math Launch is an educational game featuring high-res graphics, music and sound. The aim of the game is to build a rocket and launch it by answering multiplication questions. There is also a choice of three skill levels.

John Pospisil
Bulli NSW

```
85    CLS
7     CHAR63,0000003C003C
10    GOTO1000
506   NEXT
1000  GOSUB9000
1010  GOSUB9800
1015  CLS
1020  PRINT"WHICH SKILL LEVEL?"
1030  PRINT
1040  PRINT"    (A) VERY EASY"
1050  PRINT
1060  PRINT"    (B) INTERMEDIATE"
1070  PRINT
1080  PRINT"    (C) TOUGH"
1085  FORN=1TO10
1086  PRINT
1087  NEXT
1090  IFPEEK(29)=1THEN1100
1095  GOTO1090
1100  IFPEEK(25)=17THENA=7
1110  IFPEEK(25)=6THENA=15
1120  IFPEEK(18)=72THENA=40
1130  FORN=1TO20
1140  PRINT
1150  NEXT
1160  IFA=0THEN1020
1500  REM SREEN
1501  CHAR180,FFC3A59999A5C3FF
1502  FORN=22TO23
1503  FORM=14TO24
1504  PLOTN,M,180
1505  NEXT
1507  NEXT
1510  FORN=25TO27
1520  PLOTN,24,200
1530  NEXT
1540  FORN=18TO20
1545  PLOTN,24,252
1546  NEXT
1550  PLOT19,22,250
1555  PLOT19,23,251
1580  POKE219,208
1590  POKE218,228
1600  PRINT"LET'S START BUILDING !"
1610  FORN=1TO250
1620  NEXT
1630  POKE218,228
1640  PRINT"
1650  U=0
1800  REM QUESTION/ANSWER
1805  IFU=11THENGOSUB4300
1810  B=RND(A)
1820  C=RND(A+5)
1830  POKE219,208
1840  POKE218,230
1850  PRINTB;"  ";C;
1861  INPUTT$
1862  D=VAL(T$)
1870  IFD=C*BTHEN2000
1880  SOUND2;4,5;4,14;4
1890  SOUND2;3,5;3,14;3
1900  SOUND2;1,5;1,14;1
1910  SOUND2;4,5;4,14;4
1911  SOUND2;3,9;3,17;3
1912  SOUND2;1,7;1,16;1
1913  SOUND2;3,7;3,16;3
1914  SOUND2;1,5;1,14;1
1915  SOUND2;3,5;3,14;3
1916  SOUND2;1,4;1,12;1
1917  SOUND2;7,5;7,14;7
1920  FORN=13TO23
1930  FORM=25TO27
1950  PLOTM,N,32
1960  NEXT
1965  SOUNDN;0
1970  NEXT
1975  U=0
1980  POKE219,208
1985  POKE218,228
1986  PRINT"AH WELL,LETS START AGAIN"
1987  FORN=1TO300
1988  NEXT
1990  GOSUB5000
1995  GOTO1800
2000  REM
2005  IFU=11THEN4100
2010  GOSUB3000+U*100
2020  GOSUB5000
2030  U=U+1
2040  GOTO1800
2500  FORN=1TO256
2510  POKE4098,N
2520  FORM=1TO20
2530  NEXT
2540  NEXT
3000  PLOT25,23,96
3010  PLOT26,23,97
3020  PLOT27,23,98
3030  RETURN
3100  PLOT25,22,104
3110  PLOT26,22,105
3120  PLOT27,22,106
3130  RETURN
3200  PLOT25,21,107
3210  PLOT26,21,107
3220  PLOT27,21,107
3230  RETURN
3300  PLOT25,20,108
3310  PLOT26,20,109
3320  PLOT27,20,108
3330  RETURN
3400  PLOT25,19,108
3410  PLOT26,19,110
3420  PLOT27,19,108
3430  RETURN
3500  PLOT25,18,113
3510  PLOT26,18,112
3520  PLOT27,18,113
```

TRS-80

```
3530 RETURN
3600 PLOT25,17,113
3610 PLOT26,17,114
3620 PLOT27,17,113
3630 RETURN
3700 PLOT25,16,111
3710 PLOT27,16,111
3715 PLOT26,16,111
3720 RETURN
3800 PLOT25,15,120
3810 PLOT26,15,108
3820 PLOT27,15,121
3830 RETURN
3900 PLOT25,14,122
3910 PLOT26,14,124
3920 PLOT27,14,123
3930 RETURN
3999 END
4000 PLOT25,13,128
4010 PLOT26,13,129
4020 PLOT27,13,130
4030 RETURN
4100 FORN=18TO20
4110 PLOTN,24,32
4111 NEXT
4119 PLOT19,23,32
4120 PLOT19,22,32
4125 FORM=13TO24
4126 PLOT22,M,32
4127 PLOT23,M,32
4128 NEXTM
4130 GOSUB5000
4135 COLOR26,7,5
4140 CHAR200,FF7F7F7762
4145 POKE219,210
4150 POKE218,226
4155 POKE4098,0
4160 FORN=1TO24
4170 PRINT
4180 NEXT
4190 POKE219,209
4200 POKE218,68
4210 CHAR2,C0A0904F300C03
4220 CHAR3,000000F01010F0
4230 CHAR4,030509F20C30C0
4240 CHAR5,0000000F08080F
4250 PRINT"      WELL DONE"
4255 FORN=2TO10
4256 POKE4098,N
4260 FORM=1TO20
4270 NEXT
4280 NEXT
4290 SOUND0;0
4295 GOTO6000
4299 END
4300 FORN=1TO20
4305 POKE218,100
4310 PRINT"DATA FOR LAUNCH"
4320 POKE218,100
4330 PRINT"
4340 NEXT
4350 RETURN
5000 POKE219,208
5010 POKE218,228
5020 PRINT"
5030 RETURN
6000 SOUND26;3
6005 SOUND21;2
6010 SOUND21;1
6015 SOUND23;3
6020 SOUND21;3
6025 SOUND1;3
6030 SOUND9;3,19;3,25;3
6040 SOUND2;3,18;3,26;3
6050 FORN=1TO200
6060 NEXT
6070 PRINT"PRESS ANY KEY FOR NEW GAME"
6080 FORN=1TO10
6090 PRINT
6100 NEXT
6110 IFPEEK(29)=1THENRUN
6120 GOTO6110
8000 FORN=1TO255
8010 POKE4098,N
8020 FORM=1TO50
8030 NEXT
8040 NEXT
8999 END
9000 CHAR96,FF0F0F1F1F3F3F7F
9001 CHAR97,00FF5AA5FF
9002 CHAR250,01010103030E1E3E
9003 CHAR251;3E76F606060F0F0F
9004 CHAR1,0081422418244281
9005 CHAR252,FFFFFFFFFFFFFFFF
9010 CHAR97,FFFFFFFFFFFFFFFF
9020 CHAR98,FFF0F0F8F8FCFCFE
9030 CHAR104,FF0303030303
9040 CHAR105,FFC3C3C3C3C3
9050 CHAR106,FFC0C0C0C0C0
9060 CHAR107,000000FFFFFFFFFF
9070 CHAR108,00
9080 CHAR109,00003C3C18181818
9090 CHAR110,1800001C20180438
9100 CHAR112,2424000024242424
9110 CHAR113,00
9120 CHAR114,000000000018243C
9125 CHAR111,F0F0F0F00F0F0F0F
9130 CHAR120,0F0F1F1F3F3F7FFF
9140 CHAR121,F0F0F8F8FCFCFEFF
9150 CHAR122,0F0F0F0F0F0F0F0F
9160 CHAR123,F0F0F0F0F0F0F0F0
9170 CHAR124,FFC3C3C3FFFFFFFF
9180 CHAR128,0000000003030F0F
9195 CHAR129,003C7EFFFFFFFFFF
9200 CHAR130,00000000C0C0F0F0
9205 CHAR200,0000FFA55AFF
9210 RETURN
9800 COLOR0,13
9810 COLOR13,2,5
9820 COLOR14,2,16
9830 COLOR15,2,15
9835 COLOR16,16,5
9840 COLOR17,2,5
9845 COLOR26,2,5
9846 COLOR32,11,5
9850 FORN=6TO12
9860 COLORN,2,5
9870 NEXT
9877 COLOR1,2,5
9880 COLOR5,2,5
9885 COLOR23,2,5
9900 RETURN
9997 POKE4098,208
9998 POKE4098,0
9999 POKE4098,207
```



Apple II

CRICKET

In this game you field while the computer bats. The ball is hit from a random end in a random direction and you move around the screen, using the keyboard, to stop the ball.

If you find the game too easy, change the values in lines 165-190 from 2 and -2 to 3 and -3 (or even 4). Don't change the D values in line 170.

Type carefully because mistakes will send the program to the error handling routine and it will be hard to find the error. If you think you have an error, use the TRACE command and run the program. NO TRACE turns the TRACE off.

Good luck and have fun.

Richard de Meester
Airport West, Vic

```
5  ONERR  GOTO 1100
6  TEXT : HOME
10 L$ = "CRICKET?!"
15  HTAB 12: VTAB 15: SPEED= 240:
     FOR Z = 1 TO 9:Z$ =  MID$ (
    L$,Z,1): FOR ZZ = 32 TO 255:
     PRINT  CHR$ (ZZ);: IF  CHR$
    (ZZ) <  > Z$ THEN  CALL  - 1
    008: NEXT
20 X =  PEEK ( - 16336) -  PEEK (
    - 16336) - X =  PEEK ( - 16
    336) -  PEEK ( - 16336) - X =
     PEEK ( - 16336) -  PEEK ( -
    16336) - X =  PEEK ( - 16336
    ) -  PEEK ( - 16336): NEXT Z
    : SPEED= 255: PRINT
25  FOR I = I TO 500: NEXT : HOME
    :L$ = "BY RICHARD DE MEESTER
    , AIRPORT WEST."
26 V = 1: GOSUB 30: GOTO 55
30  POKE 34,V:ZZ =  LEN (L$) / 2:
     SPEED= 160: VTAB 23
31  FOR Z = 1 TO  LEN (L$) / 2: HTAB
    1: PRINT  MID$ (L$,ZZ + 1 -
    Z,1);: HTAB 40: PRINT  MID$
    (L$,ZZ + Z,1);: NEXT
35  SPEED= 65: FOR Z = 1 TO 23 -
    V - ZZ: PRINT : NEXT : SPEED=
    255
40  FOR Z = 1 TO ZZ: VTAB V: HTAB
    1: PRINT  MID$ (L$,ZZ + 1 -
    Z,Z);: HTAB 41 - Z: PRINT  MID$
    (L$,ZZ + 1,Z);: VTAB 24: PRINT
    : NEXT
45  FOR Z = 1 TO 20 - ZZ: VTAB V:
     HTAB Z: PRINT " "; LEFT$ (L
    $,ZZ);: HTAB 41 - ZZ - Z: PRINT
     RIGHT$ (L$,ZZ);" ";: NEXT
50  SPEED= 255: POKE 34,0: RETURN

55 V = 3:L$ = "NO COPYRIGHT <C> O
    N THIS GAME!": GOSUB 30
60  PRINT : PRINT : PRINT "      Y
    OU ARE FIELDING AT A GAME OF
          ": PRINT "CRICKET.  YO
    U ARE A BOX AND YOU HAVE TO
    ": PRINT "MOVE IT AROUND AND
     STOP THE BALL.       "
65  PRINT "      KEYS ARE:  U  I
    O": PRINT : PRINT : PRINT "
                  J     L": PRINT
    : PRINT : PRINT "
          M  ,  .": PRINT : PRINT
    : PRINT
70  PRINT "      THE FIRST ONE TO
    100 WINS": VTAB 24: PRINT "
         PRESS ANY KEY";: GET B$
75  HOME : PRINT "      THE EARLIE
    R YOU STOP THE BALL THE  ": PRINT
    "HIGHER SCORE YOU WILL GET F
    OR THE STOP. "
78  PRINT "THE BALL WILL BE HIT F
    ROM A RANDOM END  ": PRINT "
    IN A RANDOM DIRECTION.": PRINT
    : PRINT : PRINT
80  PRINT "      IF YOU ARE TOO CL
    OSE YOU WILL BE  ": PRINT "
    ASKED TO STAND FURTHER BACK.
    ": PRINT : PRINT : PRINT : PRINT
    "      PRESS ANY KEY TO START
    ";: GET B$
95  DIM BX(200): DIM BY(40)
98 TSC = 0:PSC = 0
100  GOSUB 1000: GOSUB 1020
110  GOSUB 800
120 R = 2: GOSUB 1050
130 E = X:R = 7: GOSUB 1050:D = X

140  GOSUB 800
150  IF E = 1 THEN E = 125
160  IF E = 2 THEN E = 155
165  IF D = 1 THEN DX =  - 2:DY =
     - 2
170  IF D = 2 THEN DX = 0:DY =  -
    2
175  IF D = 3 THEN DX = 2:DY =  -
    2
180  IF D = 4 THEN DX = 2:DY = 2
185  IF D = 5 THEN DX = 0:DY = 2
190  IF D = 6 THEN DX =  - 2:DY =
    2
195  IF D = 7 AND E = 125 THEN DY
     = 0:DX = 4
200  IF D = 7 AND E = 155 THEN DY
     = 0:DX =  - 4
205  GOSUB 800
206  IF XP > 50 AND XP < 200 AND
    YP < 120 AND YP > 10 THEN  HOME
    : VTAB 23: PRINT "FURTHER BA
    CK, PLEASE!": GOTO 205
210  HOME : VTAB 23: PRINT "COMPU
    TER SCORE: ";TSC;"   YOUR SC
    ORE: ";PSC
215 BX(1) = E + DX:BY(1) = 80 + D
    Y
220  HPLOT E,80 TO BX(1),BY(1)
222 G =  PEEK ( - 16336) *  PEEK
    ( - 16336)
225 I = 2
230  GOSUB 800:J = I - 1:BY(I) =
    BY(J) + DY:BX(I) = BX(J) + D
    X
235  HPLOT BX(J),BY(J) TO BX(I),B
    Y(I)
240  IF XP =  < BX(I) AND XP + 29
     >  = BX(I) AND YP =  < BY(I
    ) AND YP + 29 >  = BY(I) THEN
    242
241 I = I + 1: GOTO 230
242  IF I < 15 THEN PSC = PSC + 4
    : GOTO 245
243  IF I < 25 THEN PSC = PSC + 3
    :TSC = TSC + 1: GOTO 245
244 PSC = PSC + 1:TSC = TSC + 2
245  IF TSC >  = 100 OR PSC >  =
    100 THEN 300
246  HOME : VTAB 23: PRINT "COMPU
    TER SCORE: ";TSC;"   YOUR SC
    ORE: ";PSC
247  GOTO 100
300  TEXT : HOME : IF PSC > TSC THEN
     PRINT "YOU WON, ";PSC;" TO
    ";TSC;"!"
305  IF TSC > PSC THEN  PRINT "TH
    EY BEAT YOU, ";TSC;" TO ";PS
    C;"!"
310  IF TSC = PSC THEN  PRINT "IT
     WAS A DRAW, ";TSC;" ALL!"
```

Apple II

```
315  PRINT : PRINT : PRINT "AGAIN
      <Y/N>?";: GET A$: IF A$ = "
     N" THEN  PRINT : PRINT : PRINT
     "BYE!": END
320  IF A$ = "Y" THEN 98
325  GOTO 315
330  POKE 34,V:ZZ =  LEN (L$) / 2
     : SPEED= 160: VTAB 23
800  IF  PEEK ( - 16384) > 127 THEN
     820
810  RETURN
820  GET M$
830  IF M$ = "U" THEN XD =  - 10:
     YD =  - 10: GOTO 920
840  IF M$ = "I" THEN XD = 0:YD =
      - 10: GOTO 920
850  IF M$ = "O" THEN XD = 10:YD =
      - 10: GOTO 920
860  IF M$ = "J" THEN XD =  - 10:
     YD = 0: GOTO 920
870  IF M$ = "L" THEN XD = 10:YD =
     0: GOTO 920
880  IF M$ = "M" THEN XD =  - 10:
     YD = 10: GOTO 920
890  IF M$ = "," THEN XD = 0:YD =
     10: GOTO 920
900  IF M$ = "." THEN XD = 10:YD =
     10: GOTO 920
910  RETURN
920  HCOLOR= 0: GOSUB 1020
930 XP = XP + XD:YP = YP + YD
931  IF XP < 0 THEN XP = 0
932  IF XP > 250 THEN XP = 250
933  IF YP > 130 THEN YP = 130
934  IF YP < 0 THEN YP = 0
940  HCOLOR= 7: GOSUB 1020
945 G =  PEEK ( - 16336) +  PEEK
     ( - 16336)

950  IF XP < 170 AND XP > 80 AND
     YP < 100 AND YP > 30 THEN  GOSUB
     1010
960  RETURN
1000  HGR : HCOLOR= 7
1010  HPLOT 120,75 TO 120,84 TO 1
     59,84 TO 159,75 TO 120,75: RETURN
1020  HPLOT XP,YP TO XP + 29,YP TO
     XP + 29,YP + 29 TO XP,YP + 2
     9 TO XP,YP: RETURN
1050 X =  INT ( RND (1) * R) + 1
1060  RETURN
1100  HOME
1104  VTAB 21: PRINT "FFFF  OO  U
      U RRR  !"
1105  PRINT "F    O  O U  U R  R
     !"
1110  PRINT "FFF  O  O U  U RRR"
1115  PRINT "F     OO   UU  R  R
     !";
1116  FOR F = 1 TO 250:G =  PEEK
     ( - 16336): NEXT
1117 TSC = TSC + 4: IF TSC =  > 1
     00 THEN 300
1118  HOME : VTAB 23: PRINT "COMP
     UTER SCORE: ";TSC;"   YOUR S
     CORE: ";PSC
1120  GOTO 100
1244 PSC = PSC + 2:TSC = TSC + 2
]PR#0
```

microbee®

Australia's home grown computer